# TDDE54/TDDD87/TDIU08/725G92/9AMA73: Varför så petiga??? Är det någon vits med (detta)???

Torbjörn Jonsson <torbjorn.jonsson@liu.se>

Wed 29/09/2021 07:20

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.

> Först en liten uppdatering: Automaträttningen bör nu ge mer information om "blanktecken". Se "diff-delen". Ni kommer att se denna uppdatering vid nästa utskick (som ju sker flera gånger per dag som det är just nu). Jag har också sett att vissa av er skickar in kod flera gånger under dagarna. Smart. :-)

Detta brev riktar sig till alla! Lika mycket assistenter som er som har programmerat tidigare och givetvis er som är nya på detta med programmering.

Att förstå varför en kurs är gjord som den är är grunden till att kunna vilja göra saker som gör att man klarar av den. Jag hoppas att ni kommer att vara ytterligare ett steg på den vägen efter denna text och kanske ha en förståelse för varför vi gör som vi gör. Ta den som kvällslektyr och fundera lite på om det kanske till och med är vettigt. Om inte ge mig respons så får jag chans att göra saker bättre till senare.

OBS! Om du känner att det varit för petigt med automaträttningen tycker jag att du skall läsa detta. Det är inget som krävs för G på moment i kursen, men det kan ge tips som gör att man kanske får G lättare. :-) Det räcker att "läsa", d.v.s. du behöver inte "lusläsa". :-)

Om du tycker att allt rullar på och inga problem finns. Kolla iallafall in detta. Det är några "kul" exempel nedan som kan ge lite extra trix att ha i verktygslådan.

Jag hoppas att ni också kommer att se att vi har tankar bakom det vi gör. Inget försvarstal alltså utan bara en text som kanske ger er ännu mer förståelse och insikter i hur vi tänker. Jag tror också att du kommer att se exemplen som lite intressanta. Peppar peppar (i dubbel bemärkelse :-) ).

Om inget annat kolla iallafall in vitsen med detta. :-)

**Nu kör vi!**

Till att börja med: Nu har ni kommit så långt att jag kan använda programmeringen till att visa på vad vi är ute efter i kursen. Svårt att göra innan ni har kommit till denna punkt så säkert kan mycket frustration uppkommit beroende på detaljer, men hoppas att det inte är "för sent".

Det verkar som att det finns en del positiva saker som hänt och jag har också hört kul saker som att "Skönt att du sagt att det var jobbigaste delarna med Ada.O0 och Ada.O1. Nu känner jag att det kommer igång" eller "Det här är ju roligt. Det trodde jag inte att det skulle bli". Jättekul att höra och jag hoppas alla skall få känna så framöver.

Jag har också fått en del frågor om varför det är så noga i kursen med detaljer. De frågor jag

fått är av typen "Varför det spelar roll om man har ett blanktecken hit eller dit?" eller "Det spelar väl ingen roll om man råkat skriva ett 'a' eller ett 'å', man fattar väl ändå?" eller "Jag kan 'while' och det löser ju problemet så jag tycker att du har fel" och liknande.

Alla dessa kommentarer är helt underbart korrekta och välgrundade förstås så inget klagomål från min sida. Detta är grunden till att komma till nästa "Level". Att ifrågasätta och förhoppningsvis få en dialog som gör att man lär sig något nytt. Jag håller med om att det kan kännas petigt och att det kan kännas onödigt och det finns saker vi lagt in lite extra för att underlätta (och det kommer säkert mer som det jag fixat nu i natt), men samtidigt är det viktigt att ni hittar era egna fel i programmen och att ni läser uppgifterna (specifikationerna) noga. Hittar ni fel så påpeka det för oss så att vi kan rätta till. Vi är inte mer än människor vi heller (som tur är).

Vi har dock några problem i detta med att det är petigt, men som det finns anledning att ändå låta vara så som jag ser det. Jag skall försöka belysa detta med några exempel nedan.

Vi börjar dock med vad det är det vi försöker "lära ut" i kursen. Det är många saker, men vi tar några av dem.

1. Problemlösning.

2. Att komma på olika lösningar på samma problem och välja det som verkar bäst eller åtminstone en variant som fungerar på ett bra sätt. Att tänka "out of the box" är ofta intressant (eller "vända" på problem).

3. Programmering (kodning, felsökning, problemlösning, datahantering, ...).

4. Felsökning: Att kunna hitta sina egna fel i sin programkod (inte att vi skall hitta dem!). På tentan (eller P-uppgifterna har du bara en chans och du måste själv lösa problemet).

5. Testfall: Att själv komma på hur man skall testa sitt program och finna bra tester (inte bara det körexempel som (oftast) finns). Vi finns inte där hela tiden som ni vet.

6. Lära sig hur programspråk är uppbyggda och se likheter och skillnader mellan programspråk om man har flera i kursen. Alla programspråk är inte bra till allt!

7. ... det finns massor som sagt ...

Det finns dessutom saker som inte ens bara har med denna kurs att göra. Det glider in i saker som man kanske inte tänker på. T.ex. hur man måste ta eget ansvar för sina studier (första kurserna på LiU måste ta lite av denna "smäll" och ibland tolkas det som något negativt trots att det är bra i det långa loppet), hur man planerar sin tid (har vi inte pratat så mycket om, men ...). Saker som vi kanske inte uttryckligen pratar om, men som vi (läs "LiU") räknar med att ni kan om ett tag. Trista saker som låter som krav som vi inte borde ställa, men som ni senare kommer att tycka vara helt ok. Det är bara nästa nivå av "livet" och ni kommer att fixa det.

Jag börjar i den ände som vi kallar problemlösning för att glida mellan områdena och förhoppningsvis hänger ni med på resan och kommer i mål i slutet av texten.

Problemlösning innebär att vi skall komma fram till en punkt där ni kan få ett problem (stort eller litet) och ni kan lösa detta utan hjälp från andra (om det är små problem) eller kanske i

grupp (om det är större problem).

Var finner vi då problemlösning i "petigheten" kan man ju fråga. Det är just det som är det kluriga. Vi har små problem som gör att man måste lösa sakerna med hjälp av "små program" istället för stora problem som behöver "stora program".

**Vi tar ett exempel.**

Antag att du skall skriva ett program som skall låta användaren mata in ett heltal (låt säga att det blir talet 5). Programmet skall sen skriva ut så många A:n med mellanliggande B:n (d.v.s. i vårt exempel ABABABABA då).

> Vi börjar med en "lösning" som ni direkt kommer säga att den är dålig och det är helt rätt. Så här gör vi INTE som programmerare. Ingen av er har gjort detta heller så det är ju super!

```
Uselt program   -- Även om det finns sämre
-------------
Get(A);         -- Dåligt namnval på variabeln (eller?)
Put('A');
if A > 1 then   -- Ja det måste vi ju kolla (!)
  Put("BA");
end if;
if A > 2 then   -- Ja det måste vi ju kolla (!?)
  Put("BA");
end if;
if A > 3 then   -- Ja det måste vi ju kolla (!??)
  Put("BA");
end if;
if A > 4 then   -- Ja det måste vi ju kolla (????)
  Put("BA");
end if;
if A > 5 then   -- Ja det måste vi ju kolla (?????)
  Put("BA");
end if;
-- Ja det är bara att fortsätta om man inte ser
problemet!
```

Något som man kanske inte ser direkt är att det är sämre än det ser ut. Ser du det? Att ha nya "if" för varje test innebär att programmet måste testa ALLA villkoren oavsett om det var talet 1 användaren matade in eller om det var talet 23. Helt onödigt och alltså inte programtekniskt vettigt. Det finns "elsif" och "else" att tillgå ...

Det vi har ovan är "fullständig uppräkning" och det är "fult" i nästan alla fall och därför är detta alltså inte ok i denna kurs. Det är ju problemlösning vi skall ägna oss åt eller hur.

Alltså: Ett program som "fungerar" behöver inte vara godkänt. Däremot finns det tillfällen i kursen där vi ser mellan fingrarna på detta för att det finns annat som är i fokus. Glöm då inte att slutprodukten är att ni skall visa oss att ni kan göra "bra" program.

Jag hoppas att ni ALLA tänker: "Ovanstående program skulle jag aldrig i mina vildaste fantasier skriva!". Då har ni tänkt rätt, men det kanske inte blir så lätt att leva upp till senare. Vi får se. :-)

Vi går vidare med ett par lite bättre program/lösningar.

> I detta fall finns problemlösning i att man skall komma på hur man INTE får ut det sista B:et. Hur man gör detta kanske är på olika sätt vi kan ge två exempel på lösningar (bara den del som är relevant):
>
> ```
> Program 1
> ---------
> Get(Antal_A);
> for I in 1 .. (Antal_A - 1) loop
>   Put('A');
>   Put('B');
> end loop;
> Put('A');
>
> Program 2
> ---------
> Get(Antal_A);
> for I in 1 .. Antal_A loop
>   Put('A');
>   if I < Antal_A then
>     Put('B');
>   end if;
> end loop;
> ```
>
> Två helt olika lösningar som båda är korrekta. Är båda lika bra? I detta fall kanske det inte spelar så stor roll, men om någon hade skrivit följande program istället då?
>
> ```
> Program 3
> ---------
> Get(Antal_A);
> for I in 1 .. Antal_A loop
>   Put('A');
>   Put('B');
> end loop;
> ```
>
> Ni säger direkt: NEJ!!! Det är ju helt fel. Så kan man inte göra. Det ger ju fel resultat.
>
> Eller hur? Vi får ju inte ABABABABA utan ABABABABAB som resultat.

Här ser vi att det är väldigt lite som skiljer de tre programmen åt. Två av dem fungerar och ett ger fel resultat. D.v.s. det sista borde väl inte bli godkänt. Det tror jag att ni alla håller med om. Om inte så har vi ett "litet problem" och det är ju isåfall till för att lösas så jag hoppas att vi inte har det så att vi kan gå vidare.

Nu antar jag att vi alla är överens om att problemet ovan ger en icke godtagbar lösning (program 3). Det gör att vi har två "godkända" och en "komplettering" ovan och ni inser att detta självklart kan automatiseras och att vi kan finna detta fel genom att prova programmen mot ett

"facitprogram".

Nu finns det givetvis fall där det ställer till det så att automaträttningen inte kan göra ett perfekt jobb, men ibland får man köpa att det ändå gör att man kan få saker lösta som annars skulle ge upphov till för mycket tid att hantera (i detta fall för assistenterna) och vi kanske hittar andra saker i detta som gör att det till sist är positivt.

**Modifierad uppgift**

Vad händer nu om vi ändrar uppgiften lite. Vi säger att programmet skall skriva ut lika många A:n, men det är inte B:n som skall skrivas ut utan blanktecken. Precis det vi har som testfall i våra olika uppgifter som ni redan stött på.

Helt plötsligt är det så att man kanske vill säga att det är "orimligt" att sätta "komplettering" för det tredje programmet (där det är ett extra blanktecken i slutet av raden). Eller? Kanske det är så att det är viktigt att man är noga och att det som ingår i denna noggrannhet kanske ligger i att vi vill att ni skall hitta vägar att komma till en bra lösning istället för en som är mindre bra (eller kanske rent av fel). Vi får se.

Ett (eller två) tips: Vi har skrivit uppgifterna så att det skall gå att se hur många blanktecken som är mellan saker (redan i "pdf":en) och vi vill inte ha blanktecken i slutet av rader.

**Ett annat problem**

I våra uppgifter är det små saker som vi testar, men målet är att man skall komma på vägar som gör att man kan lösa större problem senare på ett bra sätt. Jag vill ge ett till exempel på något som vi trycker hårt på (en helt annan sak än blanktecken) som gör att man senare kan lösa problem av en annan art än den ovan. Vi har i våra uppgifter ibland saker som är likt följande exempel.

> Antag att du skall mata in ett tecken och avgöra om detta är ett 's' eller ej. Låt saga att användaren matar in tecknet och programmet skall antingen skriva ut:
>
> ```
> Innehållet i din variabel var värdet 's'.
> ```
>
> eller:
>
> ```
> Innehållet i din variabel var INTE värdet 's'.
> ```
>
> Detta beroende på vad tecknet var förstås. Följande två program är förslag på lösningar.

```
Standardprogrammet
------------------
Get(Ch);
if Ch = 's' then
  Put_Line("Innehållet i din variabel var värdet 's'.");
else
  Put_Line("Innehallet i din variabel var INTE värdet
's'.");
end if;
```

```
Ett felaktigt program
---------------------
Get(Ch);
if Ch = 's' then
  Put_Line("Du skrev in ett 's'.");
else
  Put_Line("Innehållet i variabeln var INTE 's'.");
end if;
```

Nu är det ju tveklöst så att det är helt fel i andra programmet. Det var alltså bra att läsa uppgiften noga och vara noggrann för att inte skriva fel saker. Kunde varit mindre fel, men nu var det tydliga fel denna gång (även om budskapet i resultatet var likt det vi ville ha så var det ju inte rätt).

**Koduppрepning (kodduplicering)**

Ett begrepp som vi kommer att "bråka" mer med framöver och det är "upprepning av kod". Att skriva (ungefär) samma kod på flera ställen gör att man oftast i en förlängning råkar ut för att man får problem. Exempel på detta är givetvis de inmatningar ni haft i "momstabellen" där ni har fyra (4) i princip lika inmatningsdelar där det finns två saker som har med detta att göra en liten och en mindre förstås. I Ada.O1 är det förstås ok att göra upprepning av kod till viss del, men ...

Den som är liten är att det är fyra likadana loopar och att dessa nog skulle kunna göras om så att de se precis lika ut (man måste då tänka till lite och sen behöver man underprogram som kommer lite senare så det är förstås inget som vi kräver i den uppgiften). Den mindre delen är att man som nybörjare i 99% av fallen (och de som redan programmerat en del upp emot 90% av fallen!!!) använder två Get för att lösa en inmatning av ETT tal. Låter ju som koduppрepning i mikrovariant, d.vs. precis det vi är ute efter att leka med (problemlösning kan göra detta mycket bättre än så om man har rätt "loop" och tänker till lite en extra gång). Kan man dessutom "exit when" och lite annat så blir det vackert! :-)

Att i dessa fall sitta och säga att det "spelar ingen roll" gör ju bara att man fastnar i gamla spår istället för att försöka lära sig något nytt och på det viset bli (ännu) bättre än man var. Jag hoppas att alla vill komma till nästa "Level" så jag hoppas att ni "litar på mig" att jag vet vad jag gör efter dryga 30 år med dessa frågor i dessa kurser. Om inte så tar jag gärna diskussioner med er för att se om jag kan göra annat. Jag lovar att lyssna på det ni säger.

För att återkomma till sista exemplet så kan man ju säga att det i detta exempel också finns "koduppрepning" att finna. De två texterna skiljer ju endast på det lilla att det står ett "INTE " (eller " INTE") i fallet att användaren INTE matade in ett 's'. Varför då skriva ut hela raderna (även det som är samma sak) på två ställen?

Vi ger ett ännu bättre program för att visa hur man kan lösa detta utan koduppрepning (tänk nu att detta skall motsvara STORA program och vi lär oss med hjälp av små program). Det är meningen med hela kursen. Skriv lite, men visa att du kan tänka STORT (d.v.s. generellt och programtekniskt och problemlösningsmässigt).

Den variant vi kan få till utan att göra något speciellt svåra saker och som du direkt ser att den är ok.

```
Tredje varianten av programmet
------------------------------
Get(Ch);
Put("Innehållet i din variabel var ");
if Ch /= 's' then
  Put("INTE ");
end if;
Put_Line("värdet 's'.");
```

I detta sista program har vi alltså tagit ut det som SKILJER sig från resten och har endast med detta i "if"-satsen. Mindre att skriva och alltså mindre risk för att göra fel. Jag bytte dessutom "=" till "/=" i "if" som var en detalj som belyser att det kanske går att "vända på problem" och se dem från andra håll.

OBS! Det handlar inte om att skriva så lite som möjligt utan så TYDLIGT som möjligt. Att ha bra variabelnamn (inte kryptiska saker eller förkortningar), att kanske ha en(!) mellanrad för att bryta upp långa programstycken, att stava rätt (besvräligt, oups nu stavade jag fel!) och att sen dessutom komma på ett bra sätt att lösa uppgiften är det som ingår i Ada.O0 och Ada.O1 och det är det som gör att det kan vara så jobbigt nu i bärjan av kursen (rackarns! nu stavade jag fel igen).

Nu kommer ni kanske och tycka att jag är besvärlig, men det handlar om detaljer tyvärr. Stavning är inte lätt. Vi har alla saker vi är olika bra på. Det finns alltid saker som vi kör fast på. Men det är inte meningen att detta skall vara för att "jäklas". Jag är fullt medveten om att det finns frustration i detta, men tänk då på att vi i alla lägen är ute efter att få igång er gällande problemlösningen (och allt annat).

Ni kanske tycker att det är "orimligt" att man inte får godkänt på en uppgift för att man stavat fel eller liknande. Det jag kan säga här är att jag i år har trimmat "upp" kraven på detta samtidigt som jag har "slipat" ner storleken på uppgifterna. Summan blir som jag ser det på ungefär samma nivå som tidigare. Vi har lite annat fokus, men det är samma saker vi tar upp.

Tag t.ex. i Ada.P2 (eller Ada.O2) [se under Ada.P2:s arkiv så ser ni de uppgifter som varit hittills] där ni redan vet om vad som komma skall, men där ni skall visa att ni kan komma fram till vilken typ av underprogram som är vettigt till vilken deluppgift. Varje deluppgift bör kunna lösas på ganska få rader kod och huvudprogrammet är litet.

Vi räknar med att man, när man kommit upp till "rätt" nivå för att bli godkänd på kursen (och få betyg 3 eller G, på det som motsvarar en tenta i tentaperioden enligt gamla modellen), skall klara av denna uppgift på max 30-40 minuter (låt säga 45-55 där 15 minuter extra beror på att jag räknar med att ni aldrig har programmerat tidigare då det inte finns förkunskapskrav på detta). I denna beräkning är det ca 5 minuter per underprogram och lika mycket för respektive del i huvudprogrammet samt lite tid (5-10  minuter) för att läsa uppgiften och välja rätt underprogram till respektive del.

Jag skulle säkert få klagomål från vana programmerare att jag säger så lång tid per del, men de är proffs och inte nybörjare på detta så de har fel. Ni skall ha "gott om tid" att lösa uppgifterna och därför är också tiden på P-passen 75 minuter (ordinarie tid). Första gången ni är på ett P-pass är det nog ändå stressigt, men tänk på att det finns fler P-pass och vanan kommer

allteftersom.

Ovanstående innebär att det finns åtminstone 15 minuter (lågt räknat) att titta över detaljerna. Det skall man klara av för att få sitt betyg. Jag hoppas att ni kan köpa detta när ni ser tillbaka på uppgifterna när ni blivit varma i kläderna och kommit i mål med programmeringen. Det är nog inte ett orimligt krav att ställa även om det kan kännas så innan man fått erfarenheten och nått upp till detta. Efteråt är det många som inte förstår vad som var problemet är att lösa uppgiften (t.ex. proffsprogrammerare). Ofta svårt att komma ihåg problemen man hade när man lärde sig. Se bara på hur det var når ni lärde er att cykla. :-)

Nu åter till fokus på "kodduplicering". Gör inte detta! Detta för att ju fler versioner ("kopior") av samma (eller likartad) kod man har desto fler ställen måste man rätta till om man gjort fel. Glömmer man ett ställe så är programmet fel och det är svårare att hitta felet då man ju redan rättat detta (som man tror iallafall). Det handlar alltså i det stora om att undvika att behöva göra om saker på flera ställen om man kommer på att man behöver ändra på något (det behöver ju inte vara fel som skall rättas utan det kan ju bli ändrade förutsättningar som gör att man måste ändra saker).

Nu vill jag återkomma till det som kanske några redan sett, men som är lite lurigt. Något som vårt "kära" automaträttningssystem direkt hittar, men som vi vanliga normala personer kanske missar. Jag har fört in ett stavfel i ett av exempelprogrammen ovan som gjorde att det inte skulle ha blivit godkänt. Där finns också anledningen till att man INTE skall ha duplicering av kod! Gör underprogram, loopar eller tänk på annat sätt.

Om du såg stavfelet redan när du passerade det (inte dem jag redan skrivit var felstavningar utan i ett körexempel alltså) så har du antagligen en stor fördel. Grattis! Om inte så kan jag tipsa att det var i "Ett standardprogram"-programmet. Ser du detta? Om inte så är det ännu mer viktigt att kanske sträva efter att inte göra duplicering av kod.

Nu har det blivit långt igen, men förhoppningsvis har ni fått en del småsaker att fundera på och kanske ta med er.

**Status just nu**

Lite statusrapport också: Jag såg igår ett uppsving på antalet uppgifter som skickades in och är mycket glad över detta. Fortfarande lite orolig över de som inte lämnat in något så har ni kompisar som inte är igång. Tipsa dem om att komma igång. Jag hoppas att alla som fått godkänt från automaträttningen igår också får godkänt i nästa steg. Isåfall är det 186 nya godkända uppgifter i kurserna bara från igår. Om du får komplettering är det nog på små saker som går att fixa snabbt.

**Avslutning**

Avslutningsvis vill jag säga att detta kommer att lösa sig. Bara man jobbar på och tar motgångarna på rätt sätt och inte ger efter för de små frustrationer som finns när man kör fast så blir det bra. Det löser sig bara man hjälps åt att komma igång med "tankesättet". Dessutom får man ju en massa små "kickar" när man löser problemen man har. :-)

Som alltid. Sprid glädje och hjälp oss att hjälpa er.

**Vitsen**

Får jag avsluta med en "vits" som hänger ihop lite med vår "petighet"? Jag chansar och hoppas att ni inte slår mig för den höga nivån på denna:

> A priest, a pastor and a rabbit walked into a blood donation clinic.
>
> A nurse asked the rabbit: "What's your blood type?"
>
> The rabbit replied: "I'm probably a typ o."

Ha en bra dag allihop! :-)

M.v.h.
/TJ
--

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