725G92/9AMA73/TDDD87/TDDE54/TDIU08: Rättning av Ada.P2 etc ...

Torbjörn Jonsson <torbjorn.jonsson@liu.se>

Fri 15/10/2021 13:16

To: **TDDE54_2021HT_AU <tdde54_2021ht_au@student.liu.se>; TDDD87_2021HT_FP <tddd87_2021ht_fp@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 några mail nu där studenter undrar om eller ifrågasätter rättningen av Ada.P2 och jag vill förtydliga ett par saker så att det inte är oklart även om det redan är sagt sen tidigare.

När det gäller Ada.P-uppgifterna är dessa betygsgrundande. De motsvarar alltså tentanivå på något sätt. Det innebär att vi i dessa väldigt små uppgifter tittar på saker som ni gjort i lite av ett mikroskop för att se om det är värt betyg G/VG eller 3/4/5. Vad det gäller Ada.P2 är det ju 3 eller G som gäller och då bedömer vi det på den nivån.

Uppgifterna i Ada.P2 är väldigt små som sagt och dessutom givna i upplägg. Det gör att vi tittar på saker som att "Det skall fungera!" (vilket till viss del tas av automaträttningen som ni ju har gående under hela passen) och att det är "Bra gjort ur programteknisk synvinkel." vilket innebär flera saker. Vi kan ge några exempel på detta:

> Exempel 1:
>
> Antag att man skall göra ett underprogram som skall ta fram längden på en sträng. Om man då tycker att det är en operator och att detta skall vara en "-"-operator så skall man alltså anropa denna i huvudprogrammet på t.ex. följande sätt:
>
> ```
> Len := -"Hejsan";
> ```
>
> Detta är lite speciellt då "-" alltså kan vara både en "unär" och en "binär" operator. man kan ju också använda "-" för att räkna ut differensen mellan t.ex. två tal.
>
> Oavsett detta så kommer alltså ovanstående sats att göra att "Len" får värdet 6 (om man gjort rätt i "operatorn" enligt uppgiftens definition. Detta känns i mitt tycke (och jag tror allas tycke) väldigt onaturligt. D.v.s. det är alltså fel.
>
> I detta fall är nog operator-tänket helt fel i sig, d.v.s. man borde inte ens kommit till att göra en operator.
>
> Exempel 2:
>
> Antag att det står att man skall skapa ett underprogram som skall räkna ut differensen av två heltal. Här känns det ju ganska konstigt att man skulle skapa en "+" som utför detta. Vi ser på ett anrop från huvudprogrammet igen:
>
> ```
> A := 3 + 2;
> ```
>
> Svaret skall bli 1 om man ser till uppgiften så "3 + 2" är helt onaturligt i detta fall.

I detta fall är nog operator-tänket rätt, men fel val av operator gör att det inte kan bli godkänt. Här finns det många olika operatorer också så kolla in dem och gör den som det skall vara för att matcha vad som står i uppgiften.

I fallet ovan finns det ett "litet" extra problem som gör att man kanske inte kan använda operatorn "-" heller för att lösa den uppgiften. En lite lurig sak som man tyvärr också måste förstå. Vad är då detta?

Jo. Om vi skapar en operator "-" för heltal (enligt uppgiften) så kommer denna att "köra över" den ordinarie "-" som vi redan har i Ada och det gör att vi helt plötsligt inte kommer att kunna anropa den ordinarie längre i detta program. Tänk vidare på den tanken så blir det lite otäckt.

I detta fall borde man alltså inte ens ha gjort "-" som operator då det ställer till det ytterligare. Ett väldigt besvärligt problem alltså. Skulle det inte finnas andra underprogram som kan vara operatorer i P-uppgiften får vi nog se denna som ett alternativ dock, men helst inte alltså.

Exempel 3:

Hur anropar man en operator? Jag skulle vilja referera till gymnasieskolans matematik här (borde absolut finnas med där om det inte finns i högstadiet på grundskolan). Där pratar de om "operander" och "operatorer". När du räknar ut "2 + 3" är 2:an och 3:an operander och + är operatorn.

Ovanstående är alltså något som vi borde kunna räkna med att ni kan, men jag tar upp det ifall ni glömt detta (eller att de missat att ta upp detta på tidigare stadium).

Det finns också "unära" operatorer. I Ada har vi t.ex. "+" och "-" som då står före sin operand. Dessa används för att tala om ifall det är positivt eller negativt tal som står efteråt. Om det inte står något är de underförstått positivt, men detta är ju också grundläggande på samma sätt som mycket annat. Se det inte som kritik eller något annat. Jag bara tipsar om saker som vi anser skall finnas i grunden.

I Ada "kan man" anropa en operator precis som en "funktion", MEN detta är absolut inte meningen att man skall använda bara för att det går. *[Precis som att använda "out" i funktioner vilket är en ganska knasig tanke om man går vidare lite.]*

Man kan alltså skriva ett anrop till "+" på följande sätt (inte snyggt!):

```
"+"(A, B)
```

Observera nu att det INTE är snyggt på något sätt. Man bör (läs: "skall") anropa på följande sätt:

```
A + B
```

Detta gäller förstås också "<" och andra operatorer.

Det finns en massa andra "små" saker som vi inte vill se i P-uppgifter också. Jag "rabblar" upp ett gäng av dessa, men det är inte en fullständig lista.

- Användning av "globala variabler" i underprogram. Tips: Skriv inte variablerna ovanför underprogrammen så slipper ni risken för att missa ett "godkänt".

- Kodduplicering. Även på "mikronivå". Läs mail om detta sen tidigare.

- Dålig indentering ("felaktig" inskjutning av kod).

- Dåliga identifierarnamn. Tänk på att det du skriver läses av andra och de skall förstå och inte ta illa upp heller (som att t.ex. skriva "Idiot" som variabelnamn säger mer om den som skriver det är något annat eller hur). Förlåt att jag blev "pappa" nu ...

- Att man inte vet hur man använder enkla programkonstruktioner (såsom att skriva "if A = True then", som ni ju antagligen ser vad det är galet med om ni tänker på vad en operator "=" gör ...).

- Fullständig uppräkning. Även på låg nivå. D.v.s. vi vill inte ha:

```
if A = 1 then
  ...;
elsif A = 2 then
  ...;
elsif A = 3 then
  ...
end if;
```

  Om det inte är omöjligt att komma ifrån detta. Tänk underprogram, loopar (och senare "fält"). Det finns mycket som går att undvika.

- ...

OBS! Vissa av dessa ser vi kanske mellan fingrarna på i Ada.P2, men ju högre betyg man är ute efter desto mer vill vi se att det ser "bra" ut. Tänk på att det är ni som skall visa oss vad ni kan och inte att vi skall tala om för er vad ni skall göra för att lösa problemen. Vi ger därför lite "vaga" återkopplingar ibland, men det är för att ni skall diskutera och prata med varandra och assistenterna. Det är en del av kursens mål.

Jag hoppas nu att ni förstår att det inte är så konstigt att vi sätter "komplettering" (vilket betyder kom tillbaka på nästa pass och lös en ny uppgift vad det gäller P-uppgifter) för saker som är "osnygga" eller som vi uttrycker det "programtekniskt inte bra". Det handlar till sist alltid om att vi skall värdera om det är tillräckligt bra för att nå upp till betyg på kursen.

Ha nu en trevlig helg och hoppas att detta gav er ytterligare chans att klara uppgifterna snabbare.

M.v.h.
/TJ
--

_/_/_/_/_/_/_/_/ ***Torbjörn Jonsson***

_/ _/ ***013-28 24 67***

_/ _/ ***Torbjorn.Jonsson@LiU.SE***

_/ _/ _/ *IDA/SaS/**UPP***

_/ _/_/ ***Institutionen för Datavetenskap***

- - - - - - - - - - ***Linköpings universitet***