DD1328 kursanalys vt24

DD1328 Grundläggande datalogi för tekniska beräkningar 9,0 hp
Analysen är gjord av Hedvig Kjellström den 2 juni 2024.

Kursen, som ges för CTMAT årskurs 1, genomfördes för första gången under period 3 och 4, och ersatte från i år DD1320 6 hp och DD1396 3 hp (men samläste med DD1396). 63 studenter var registrerade, och 52 studenter har slutfört kursen hittills (varav 31 A, 13 B, 3 C, 2 D och 3 E).

Återkoppling om kursen inhämtades på följande sätt:

- **Två möten under kursens gång mellan examinator Hedvig Kjellström och kursnämnden bestående av två studenter ur årskursen,**
- **Onlineenkät i slutet av kursen, där studenterna (17 svar) fick lämna fritextsvar på följande två frågor "Nämn en eller flera saker i kursen som du tycker ska behållas som de är till nästa år." och "Nämn en eller flera saker i kursen som du tycker ska förändras till nästa år - och föreslå på vilket sätt de ska förändras.",**
- **Kursutvärderingsmöte mellan examinator Hedvig Kjellström, Ric Glassey (kursledare DD1396 som utgör en modul i denna kurs), Emma Enström (kursledare DD1327 som är Fysiks motsvarighet), handledare och assistenter (varav en närvarande) och kursnämnd (varav en närvarande).**

Organisation och lärandeaktiviteter

Kursen är uppdelad i fyra moduler: A, B, C och Parallellprogrammering. Modulen Parallellprogrammering samläser med DD1396 och utvärderas som en dela av den kursen. I denna analys fokuserar vi på modulerna A-C.

Föreläsare, kursledare och examinator var Hedvig Kjellström. Övningsassistenter var Nils Forsgren och Christian Lindeborg. Datorsalshandledare var Nils Forsgren, Christian Lindeborg, Anna Mårtensson, Oskar Rune, Johan Tell, Ryll Åman och Frederik Spang Dyhrberg Nielsen.

Var och en av modulerna A-C examinerades med en hemuppgift (se mer detaljer nedan) och bestod av följande moment:

- 4 föreläsningar om 2x45 minuter
- 2 övningar om 2x45 minuter
- 2 datorsalspass för hjälp med hemuppgift om 2x45 minuter
- 2 datorsalspass för muntlig redovisning av hemuppgift om 2x45 minuter

Inför varje föreläsning fick studenterna 1-3 korta texter att läsa (onlinematerial från Stefan Nilsson[1] som är framtaget för Fysiks kursmotsvarighet DD1327 samt texter från GeeksforGeeks[2] som har korta, pedagogiska genomgångar av datalogibegrepp). Föreläsningarna innehöll diskussionspunkter och problemlösning för att aktivera studenterna

[1] https://yourbasic.org/algorithms/
[2] https://www.geeksforgeeks.org/

enligt metoden Structured Lecturing[3]. I början gick ungefär 50 personer på föreläsningarna, mot slutet ungefär 30 personer.

Övningarna bestod i arbete med uppgifter som var snarlika de i modulens hemuppgift. De var till en början organiserade så att båda assistenter löste uppgifterna i dialog med klassen. Efter tre övningar ändrade vi så att i ena gruppen arbetade studenterna självständigt med uppgifterna och frågade om hjälp när det behövdes. Detta var mycket uppskattat. I början gick ungefär 40 personer på övningarna, mot slutet bara runt 10 personer. Enligt kursnämnden hade tyvärr många slutat gå när ändringen infördes.

Återkoppling om organisation och lärandeaktiviteter

Innehåll (enkät och kursnämnd): Många studenter tyckte innehållet var intressant. Några tyckte kursen var alldeles för lätt, de kunde redan det som lärdes ut (*alltså uppfyllde redan lärandemålen, HK anm*). Andra tyckte kursen var lagom svår, och några tyckte att den var för omfattande. HK tror att detta främst beror på att studenterna har väldigt olika förkunskaper i datalogi och programmering (till skillnad från till exempel matematik, där nästan alla studenter läst samma gymnasiekurser innan KTH). Kursens lärandemål och förkunskapskrav är avpassade för dem utan programmeringserfarenhet innan KTH. Det är värt att notera att alla dessa studenter har en mycket stark allmän studiebakgrund och att ingen student hade stora problem att följa kursen.

Föreläsningar (enkät): Några studenter tyckte föreläsningarna gick för långsamt fram och gick igenom för lätta saker. Andra tyckte att de var bra med lagom tempo. Samma sak här, föreläsningarna är framtagna för kursens lärandemål och förkunskapskrav, som i sin tur är avpassade för dem utan programmeringserfarenhet innan KTH.

Teori (enkät och kursnämnd): Genomgången av teoretiska begrepp som tidskomplexitet och rekursion i början av kursen var för snabb och ytlig, tycker sig HK utläsa - studenterna säger inte det direkt, men det har varit många frågor och klagomål på kraven på komplexitetsanalys. Det visade sig också att många har luckor i förkunskaperna om vissa begrepp som tex induktion och objektorientering.

Övningar (enkät och assistenter): En student nämnde specifikt övningarna som något bra och givande, och ingen tycker att de ska förändras. Assistenterna och HK tycker dock att det skulle vara bra att låta en av grupperna jobba självständigt redan från början nästa år.

Handledning (kursnämnd): Studenterna är nöjda med den individuella hjälp som erbjudits.

Planerade förändringar i organisation och lärandeaktiviteter

Innehåll: Inga förändringar.

Föreläsningar: HK vill fortsätta att rikta föreläsningarna till de studenter som inte har extra förkunskaper. För att ge guidning till de studenter som redan uppfyller vissa lärandemål, kommer varje föreläsning föregås av ett quiz (i Canvas) som visar om man kan det som ska gås igenom. Om man får godkänt på quiz:et behöver man inte gå på föreläsningen. Detta enligt metoden frågebaserat lärande[4].

---

[3] Biggs and Tang. *Teaching for quality learning at university*. Open University Press, 2011.
[4] https://cocreate.idocos.eu/books/torus-pa-kth/page/vad-ar-fragebaserat-larande

Teori: Mer tid kommer läggas på de inledande föreläsningarna på att gå igenom teoretiska begrepp, främst tids- och utrymmeskomplexitet, men även induktion och rekursion. Vi kommer även att repetera begreppen vid 1-2 senare tillfällen i kursen.

Övningar: I en eller båda grupperna kommer studenterna att arbeta självständigt under handledning av assistenten. Nytt är att de arbetar i grupper om 3-5 istället för individuellt.

Examination

Varje modul A-C examineras med en hemuppgift (en datorlaboration med både programmerings- och teoriinslag). Den består av sju programmeringsuppgifter, tre obligatoriska och fyra valfria. Man samlar till det betyg man vill ha - en valfri uppgift ger D, två valfria ger C osv. Programspråk är primärt Python, men studenterna är fria att använda Go, Java, Matlab eller C++ om de vill öva på dessa språk och känner sig trygga med att lösa praktiska programmeringsutmaningar själva.

Hemuppgiften redovisas med tre moment:

- Inlämning i Git av källkod för alla lösta uppgifter.
- Inlämning i Canvas av labbrapport i pdf-format. Labbrapporten följer ganska delaljerad specifikation som ges i labblydelsen. Under Modul A delade HK ut en mall. Rapporten kan skrivas i valfritt program som Word, Latex eller GoogleDoc.
- Muntlig redovisning. Redovisningen sker framför datorn under 15 minuter tillsammans med en datorsalshandledare.

Försenade inlämningar är tillåtna, men betyget sänks ett steg per vecka som inlämningen är försenad.

Obligatoriska uppgifter som blir underkända kompletteras i efterhand, och kompletteringen räknas som en veckas försening, dvs betyget sänks ett steg. Man kan däremot inte i efterhand komplettera frivilliga uppgifter - när de underkänns räknas de bort och betyget sjunker ett steg per underkänd uppgift.

Slutbetyget i kursen är ett medelvärde av betygen för de tre hemuppgifterna A, B och C. Parallellprogrammeringsmodulen ger endast P/F och inverkar inte på kursens slutbetyg.

Återkoppling om examination

Examinationsformat (enkät): Detta är ju det etablerade formatet för examination i grundläggande datalogikurser så det är inte en stor diskussionspunkt, men HK vill lyfta fram att många studenter i enkäten tyckte att examinationsformatet med hemuppgifter var bra.

Skriftlig redovisning (enkät, lärare och assistenter): Inlämningen i rapportform har fördelen att teorimoment kan examineras mer transparent än med bara muntlig redovisning, och att studenterna övar sig i att skriva skriftliga rapporter. Dock behöver formatet ses över av tre anledningar:

- För det första så tycker många studenter både i enkäten och som HK pratat med att det är svårt, tidskrävande och otydligt med redovisningen i rapportform.
- Många studenter tycker också att begreppet pseudokod är oklart och har problem att abstrahera bort programspråksspecifika saker.

- Slutligen poängterar assistenterna att det ofta finns brister i strukturen på studenternas programkod, som inte rättas skriftligt eftersom den inte ingår i rapporten utan lämnas in separat.

HK noterar att man en assistent att 15 minuter är för kort tid för redovisning när studenten gjort alla uppgifter. HK kan instämma i detta.

Regler för betygssättning och komplettering (enkät och kursnämnd): Många studenter kommunicerade att de kände sig stressade över reglerna för betygssättning och det faktum att man inte får komplettera frivilliga uppgifter. HK noterar att själva betygssystemets design följer standard på EECS kan slå flera flugor i en smäll, se nedan.

Muntlig redovisning (assistenter): De muntliga redovisningarna fungerade bra (inga specifika studentkommentarer) och fyllde sitt syfte, att visa på studentens förståelse av uppgiften. Dock tycker - vi tillåter som regel inte betygshöjande komplettering av labbuppgifter. Däremot bör vi se över formuleringen av labblydelser och mallar för rapporten, så att det tydligare framgår i förväg vad som krävs för godkänt, se nedan.

Planerade förändringar i examination

Examinationsformat: Inga förändringar, tre betygssatta hemuppgifter A, B och C nästa år.

Skriftlig redovisning: Vi föreslår att rapporten nästa år skrivs i Markdown-format[5] och integreras med källkoden för uppgifterna. Detta har ett flertal fördelar:

- För det första är det en mindre konceptuell tröskel för studenterna än att skapa och strukturera ett nytt dokument i Word eller Latex - det är en naturlig fortsättning på att skriva kommentarer i källkoden.
- För det andra är Markdown ett mycket användbart verktyg som studenterna har nytta av att kunna, inte minst tillsammans med Git.
- Vi kan också då utelämna momentet att formulera pseudokod, som är en tröskel för många. Python är designat för att vara så likt pseudokod som möjligt, och vi kan låta studenterna beskriva sina algoritmer i Python, genom att helt enkelt redovisa sin källkod i rapporten.
- Detta ger också fördelen att källkoden kommer att granskas, vilket gör att felaktigheter upptäcks redan på det skriftliga rättningsstadiet.

Man slår alltså flera flugor i en smäll genom att övergå till Markdown. Inlämning kommer att ske i Git.

Muntlig redovisning: Vi kommer att ge längre tid för den muntliga redovisningen ju fler uppgifter som ska redovisas.

Regler för betygssättning och komplettering: Vi kommer att se över lydelser för alla tre hemuppgifter så att det är glasklart vad man måste göra på varje deluppgift för att bli godkänd. Vi kommer också att göra en mer detaljerad mall för en Markdownrapport.

[5] https://www.markdownguide.org/