Kajsa Bråting, Lennart Rolandsson & Cecilia Kilhamn

# Vad hände med programmeringen?

Här redovisas resultat från ett fyraårigt forskningsprojekt om implementeringen av programmering i skolans matematikundervisning. Både lärare och läromedel har studerats och resultaten väcker frågor om satsningen varit så genomtänkt som man hade kunnat önska.

I takt med det digitala samhällets framväxt har regeringen haft stora ambitioner med att öka elevers digitala kompetens, vilket resulterade i en reviderad läroplan som trädde i kraft under 2018. Intentionen med den nya läroplanen var att elever skulle utveckla en förståelse för hur digitalisering påverkar såväl samhälle som individ. Våra elever behöver kunna använda digitala verktyg och ha ett kritiskt och ansvarsfullt förhållningssätt till media och information. Som en del av den digitala kompetensen fördes programmering in i ämnena matematik och teknik genom grundskolans alla stadier.

Det är inte första gången som datorer dyker upp i svenska skolans läroplaner. Redan i 1969 års läroplan återfinns ”orientering om datamaskiner” som ett av matematikämnets huvudmoment för högstadiet. Även om fokus då var mera riktat mot datorn i sig gjordes innovativa försök att undervisa i programmering, framförallt på gymnasienivå. När 1980 års läroplansreform sjösattes infördes istället ”datalära” som ett nytt innehållsområde i matematik och samhällskunskap i grund- respektive gymnasieskolan och ”datakunskap” även som ett eget ämne i gymnasieskolan. Datalära fick dock inget större genomslag då skolornas tillgång på datorer var bristfällig och lärarna saknade fortbildning. I samband med 1994 års läroplansreform försvann såväl datalära som programmering från grundskolans kursplaner. Det var inte förrän 2018 som programmering återinfördes, denna gång som en del av det vidare begreppet *digital kompetens* inom ämnena matematik och teknik. Frågan vi ställer är: Har det gått bättre den här gången?

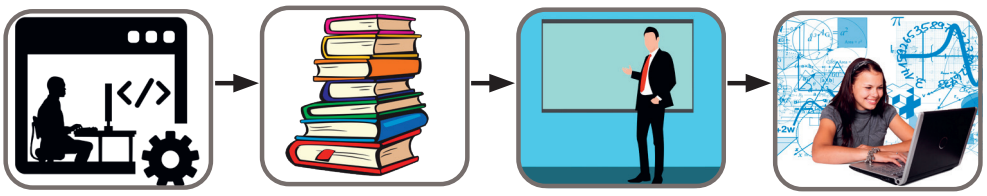
## Forskningsprojektet inPAC

Forskningsprojektet inPAC, *Integrating programming in school mathematics – exploring the intersection of algebraic and computational thinking*, startades 2019 och kommer att avslutas under 2023. Syftet med projektet har varit att undersöka vilken initial påverkan 2018 års implementering av programmering har haft på den svenska skolmatematiken. Noterbart är att Sverige har valt att integrera programmering i nära anslutning till området algebra, vilket är unikt i ett internationellt perspektiv. Det svenska fallet gjorde det möjligt för oss att undersöka hur datalogiskt tänkande relaterar till algebraiskt tänkande samt hur denna kombination skulle kunna skapa nya möjligheter för grundskoleelever att lära sig matematik.

Projektet har varit uppdelat i två större delstudier där den ena bestått av analyser av det nya programmeringsinnehållet i läromedel och material på webben, medan den andra utgjorts av intervjuer med lärare i grundskolan samt analyser av lesson studies där lärare planerat och genomfört lektioner om programmering i matematik. Ett återkommande tema i projektet har varit att undersöka hur semantiska och syntaktiska likheter och skillnader inom matematik och programmering kan påverka elevers lärande i matematik. Här nedan kommer vi att presentera några av de viktigaste resultaten från projektet och de funderingar dessa väckt.

## Vem bestämmer innehållet?

Projektet inPAC utgår från den franske didaktikforskaren Yves Chevallards teorier om transposition av kunskap. I korthet handlar teorin om att kunskap förändras och anpassas (transponeras) när den flyttas mellan olika instanser i samhället. Eftersom införandet av programmering initierades av regeringen kan det beskrivas som en top-down-process. Bilden nedan illustrerar hur programmeringskunskap rör sig från *yrkeskunskap*, via *avsedd kunskap* till *undervisad kunskap* och slutligen finns som *tillgänglig kunskap*.



**Yrkeskunskap**
Den kunskap som finns hos programmerare och programutvecklare.

**Avsedd kunskap**
Den kunskap som framträder i läroplan, läromedel och lärares planeringar.

**Undervisad kunskap**
Den kunskap som läraren förmedlar i klassrummet.

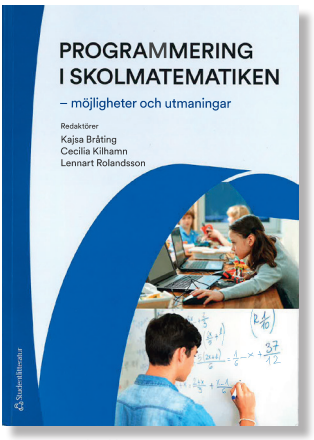
**Tillgänglig kunskap**
Den kunskap som eleverna lär sig.

Chevallard menar att i varje instans förändras kunskapen. Några exempel på beslut som påverkar och transponerar programmeringskunskapen är i vilka ämnen innehållet placeras, vilka ord som används för att beskriva mål och innehåll, vilka programmeringsspråk och miljöer som förespråkas, vilka uppgifter som väljs och hur dessa sekvenseras samt vilken datorteknik som görs tillgänglig för elever. I projektet har vi framförallt fokuserat på instanserna avsedd kunskap och undervisad kunskap samt de två första transpositionerna, det vill säga de två första pilarna i bilden.

### *Från yrkeskunskap till avsedd kunskap*

Anthemis Raptopoulou har i sin avhandling studerat den första transpositionen från yrkeskunskap till avsedd kunskap så som den formulerats i den reviderade läroplanen 2018. Vi har försökt förstå varför programmering placerats i matematikämnet och varför det anses tillhöra det centrala innehållet algebra. Det har varit svårt att finna motiveringar till detta, och Raptopoulou skriver i sin avhandling att läroplanen saknar en definition av begreppet programmering och att det ”fortfarande [är] oklart vilka pedagogiska eller demokratiska frågor programmering förväntas lösa” (s 123). Inom projektet

har vi studerat nästa fas av transpositionen då läroplanen tolkats av läroboksförfattare och av lärare som planerar sin undervisning. Där är det tydligt att denna oklarhet kvarstår, både vad gäller definitionen av och målsättningen för programmering i matematik. Läs mer om detta i antologin *Programmering i skolmatematiken – möjligheter och utmaningar.*




I ett samtal med en person som deltog i revideringen av matematikkursplanen undrade vi varför programmering placerats i innehållsområdet algebra. Svaret vi fick var att programmering skulle få alltför stor tyngd om den gjorts till ett eget centralt innehåll, och att bland de befintliga centrala innehållen var algebra det ”minst dåliga”. I våra ögon har detta visat sig vara oövertänkt och kan ifrågasättas. Lärare i våra studier talar mer om programmering i termer av logiskt tänkande och problemlösning än som algebra (Kilhamn m fl, 2021).

Även om det finns en hel del som överlappar så finns det också stora skillnader mellan algebra och programmering. En skillnad är att *datalogiskt tänkande* (computational thinking) skiljer sig från *algebraiskt tänkande*. De allra flesta länder som infört programmering i skolan har utgått från en önskan om att utveckla elevernas datalogiska tänkande. I den svenska kursplanen för matematik nämns inte datalogiskt tänkande. Faktum är att ordet *tänkande* inte används överhuvudtaget i matematikkursplanen, vilket i sig är anmärkningsvärt. Vi ser inom projektet hur lärare brottas med detta. De uttrycker ofta att målet med undervisningen är att utveckla tänkande, frågan är i så fall – vilken sorts tänkande? En annan skillnad mellan algebra och programmering är innebörden av begreppet algoritm. I matematiken definieras en algoritm som en metod som kan användas för att lösa en klass av problem, medan det i programmering är en stegvis instruktion för att lösa ett specifikt problem. I äldre matematikkursplaner lades mycket tyngd på algoritmer för de fyra räknesätten, men i dagens läroplan återfinns det endast i relation till programmering. Vi finner det även märkligt att begreppet *algoritm* lyfts fram som centralt i matematikämnet men inte alls nämns i teknikämnet där algoritmer skapas för att styra föremål och datorer.

## *Från läroplan till lärare – vad händer på lektionerna?*

När lärare ska tolka läroplanstexten ser vi en viss vilsenhet. Det gäller såväl frågor om innehåll som syfte. Ett resultat av våra lärarintervjuer är att många söker hjälp på webben och på sociala medier. Raptopoulou visade att kommersiella aktörer har haft stort inflytande redan i den första fasen då läroplanstexten skrevs, och vi ser nu att kommersiella aktörer även har stort inflytande över innehåll och aktiviteter i klassrummet. Detta i sin tur innebär att det saknas en överblick över vad som faktiskt görs på lektionerna med stor risk för att likvärdigheten i undervisningen inte upprätthålls. Våra resultat visar att det finns oklarheter kring vad som räknas som programmering och om det är viktigt att särskilja just programmering från annan kunskap om digitala hjälpmedel inom matematikämnet.

Då man lär sig att programmera kan skillnaden mellan programmeringsmiljö och programmeringsspråk upplevas obefintlig, speciellt i visuella miljöer såsom Scratch. För textbaserade språk är skillnaden mellan miljö och språk tydligare eftersom man växelvis arbetar med koden och provkör den.

På så vis får man också en djupare förståelse för hur datorn tolkar och utför givna instruktioner. I Scratch arbetar man med färdiga block och behöver inte kontrollera kodens syntax. Då ligger fokus mer på en logisk sekvens av instruktioner som påminner om en berättelse.

På 70- och 80-talet, då datorskärmarna inte erbjöd högupplöst grafik, fanns en naturlig uppdelning mellan programmering på papper och på dator. Arbetssättet avspeglar sig i läroböcker från den här tiden där flödesdiagram först arbetades fram på papper och sedan som kod på datorn. Elevers problemlösning skedde därför i två steg: först den generella logiska problemlösningen på papper och sedan den specifika problemlösningen med kod på datorn. I den första fasen kunde matematiska problem hanteras, medan den andra fasen fokuserade rent programmeringstekniska problem. Med dagens snabba datorer arbetar elever iterativt och erbjuds därför inte samma möjligheter att upptäcka om felet är syntaktiskt eller logiskt, eller rent av ett körfel som beror på att man arbetar mot en molntjänst på nätet. Det här resulterar lätt i röriga lektioner där varken elever eller lärare förstår orsaken till felen som uppstår och problemlösning blir svårt. För att lyckas med undervisningen behöver lärare kunna hantera såväl den specifika programmeringsmiljön som de nya arbetssätt som kan behövas. Mot denna tekniska kunskap ställer vi oss frågande till varför Skolverket i den senaste revideringen 2020 valde att plocka bort programmering från innehållsområdet problemlösning i kursplanen för matematik.

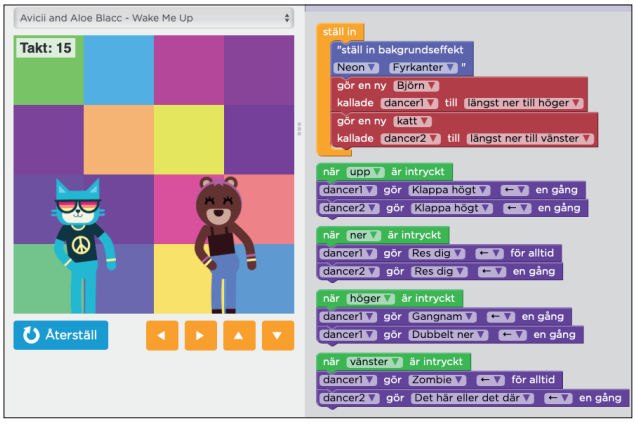
Dagens tydliga implementering i form av en top-down-process, samt den stora tillit som råder till information från sociala medier och mer eller mindre kommersiella aktörer på nätet, ter sig ganska märklig. Under 1980-talet kan vi se att datavetenskapliga strukturer och programmeringsspråk diskuterades mera explicit av skolmyndigheter och lärartidskrifter. Man kan därför fundera över huruvida implementeringen av programmering i matematik skulle få en större acceptans bland lärare om datavetare, ämnesdidaktiker och lärare tillsammans hade utvecklat ett undervisningsmaterial med lektioner och undervisningsaktiviteter.

## Programmering i matematiken – en bra ide?

Att programmering placeras inom ramen för matematikämnet är, som vi tidigare nämnt, inte en självklarhet. I Danmark ägnar man fortfarande tid åt hur programmeringen ska föras in i läroplanen genom att prova två olika alternativ där en integrering av programmering i matematikämnet är ett av dem. Om programmering istället läggs i ett nytt ämne, som exempelvis *Computing* i England eller *Teknologiförståelse* som är det andra alternativet i Danmark, då behövs direktiv för lärarbehörighet i detta ämne med ändrat utbud i lärarutbildningen som följd, vilket inte gjorts i Sverige. Här var man istället snabba med att skriva in programmering i läroplanens matematikinnehåll vilket tillsammans med otillräckliga fortbildningsresurser har inneburit att matematiklärare på alla stadier undervisar i programmering oavsett om de kan programmera eller ej. Skolverket har visserligen genomfört lärarfortbildningar vid svenska universitet och sammanställt ett utbildningsmaterial på sin lärportal men satsningarna har tyvärr endast omfattat en mindre lärargrupp. Visst går det att dra nytta av programmering i matematikundervisningen, något vi själva testat på och skrivit om i artikeln *Variables in*

*early algebra – exploring didactic potentials in programming activities*. Men för att göra det krävs omfattande fortbildning kring hur man designar aktiviteter som innehåller viktiga matematiska idéer samtidigt som de fungerar som bryggor mellan matematik och programmering. I våra intervjuer framkommer att många skolor varken har råd eller tid att fortbilda sina lärare i programmering. Inom inPac-projektet ställer vi oss därför frågande till om det svenska tillvägagångssättet verkligen är förenligt med en skola som ska bygga på forskning och beprövad erfarenhet.

När vi studerat programmering i läromedel och lektionsplaneringar sluter vi oss till att undervisning i både programmering och matematik förlorar på den starka kopplingen. Vi ser att det är utmanande att skapa ett matematiskt intressant innehåll i programmeringsaktiviteter innan man kan tillräckligt mycket matematik. Framförallt i de lägre åldrarna är det lättare att hitta på meningsfulla aktiviteter i andra skolämnen, såsom dansprogram (musik), animerade sagor (svenska/bild/teknik), styrning av minirobotar och blinkande ljus (teknik) eller instruktioner för att baka en kaka (hemkunskap). Matematiken lyser med sin frånvaro samtidigt som detta görs på matematiklektioner eftersom programmering enligt kursplanen hör till matematik.

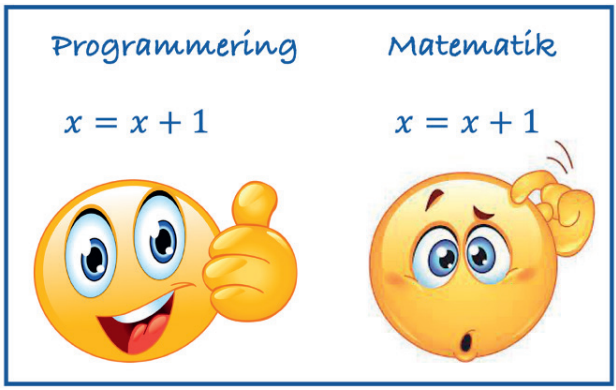



*Dansprogram med blockprogrammering – kul programmering men utan matematik.*

I analyser av matematikläromedel har vi undersökt vilken roll det nya programmeringsinnehållet spelar i relation till matematiklärandet (Bråting & Kilhamn, 2021). Här kan man tänka sig att programmeringen skulle kunna användas för att effektivisera beräkningar, exempelvis när man ska kasta en tärning 1000 gånger för att beräkna sannolikheter. Man skulle också kunna tänka sig att programmering används som ett verktyg för att utforska matematik, till exempel genom att undersöka innebörden av variabler, funktioner eller primtal. Resultat visar dock att programmeringen sällan används på dessa sätt i läromedlen, istället fungerar ett ganska simpelt matematikinnehåll enbart som en kontext för att lära sig programmera. En vanligt förekommande uppgift är att eleverna ska skriva ett program så att datorn ritar en kvadrat för att lära sig stegvisa instruktioner och loopar. Det är förstås inget fel med en sådan uppgifter men om den ges i ett matematikläromedel för årskurs 6 bidrar det knappast till elevernas matematiklärande.

Vidare ser vi inom projektet flera begreppsliga fallgropar som kan uppstå av denna integrering. Begrepp som algoritmer, likheter, variabler och funktioner har inte alltid samma innebörd inom algebra och programmering. Exempelvis representerar likhetstecknet en tilldelning inom programmering, medan det inom matematiken representerar en relation mellan två objekt som är identiska. Detta innebär bland annat att likheten $x = x + 1$ får olika innebörd inom matematik och programmering. I matematiken är likheten ganska meningslös eftersom den inte är sann för något värde på $x$. Men i en programmeringskod är den helt gångbar eftersom likhetstecknet representerar tilldelningen ”addera 1 till värdet $x$”, vilket ofta används när ett program ska loopa igenom ett antal konsekutiva heltal.

Ett annat exempel är begreppet funktion som i en matematisk kontext handlar om ett samband mellan variabler men som samtidigt i det nyinsatta programmeringsinnehållet i matematikläromedel kan definieras som instruktioner för att lösa problem. I de läromedel vi analyserat har begreppsliga skillnader mellan programmering och matematik inte behandlats.




Även programmeringsundervisningen begränsas av att låsas fast i en matematisk kontext. Många programmeringsuppgifter konstrueras så att de passar in i en matematisk kontext när fokus borde vara hur man lär sig att programmera på bästa sätt. I läromedelsstudierna ser vi att detta leder till att det läggs tonvikt på vissa programmeringsbegrepp som till exempel loopar, upprepningar och stegvisa instruktioner medan andra fundamentala programmeringsbegrepp som namngivning, villkor och debugging (felsökning) förekommer mycket sparsamt. Detta blir kontraproduktivt, det hade såklart varit enklare om programmeringen hade kunnat undervisas utifrån sina egna premisser istället för att gå omvägen genom matematikämnet. Våra undersökningar visar dessutom att lärare ställer sig frågande till om de ska undervisa i programmering eller matematik. Allt detta hade enkelt kunnat undvikas om programmering hade erbjudits som ett eget ämne med egna lärandemål och med lärare som var utbildade inom programmering.

Liknande erfarenheter vittar även andra om, till exempel Elisabeth Hector och Lena Thelander som genomfört ett forskningsbaserat projekt i årskurs 5–7 med fokus på porogrammeringsundervisning i matematikämnet. Läs mer om deras erfarenheter i artikeln *Kan programmering bidra till lärande i matematik?*.

## Slutord

När vi startade projektet trodde vi att skolmatematiken skulle komma att förändras en del i samband med programmeringens inträde och de praktiker som utgör programmerarens vardag. Kanske skulle matematikundervisningen komma att präglas av ett mera undersökande arbetssätt inspirerat av trial and error och tinkering, till exempel genom att elever utforskar matematiska samband och löser matematikproblem med hjälp av att testa, förändra och debugga. Men detta har inte (ännu) hänt och vi ser heller ingen tendens till att det kommer att hända.

Som framgått av den här texten pekar våra resultat mot att varken matematik- eller programmeringsundervisningen i Sverige har gynnats av införandet av programmering i skolmatematiken. Trots detta vill vi understryka att vi är positiva till att programmering förs in i skolan. Men för att satsningen ska lyckas behöver processen vara noga genomtänkt i alla delar av transpositionen; från programmerare i samhället via styrdokument och vidare till läromedel, lärare och slutligen eleverna i klassrummet. Det tycker vi inte att den svenska satsningen har lyckats med.


LITTERATUR:

Bråting, K., & Kilhamn, C. (2022). The integration of programming in Swedish school mathematics: investigating elementary mathematics textbooks. *Scandinavian Journal of Educational Research*, 66(4), 594–609.

Bråting, K., Kilhamn, C. & Rolandsson, L. (red) (2021). *Programmering i skolmatematiken – möjligheter och utmaningar.* Studentlitteratur.

Hector, E. & Thelander, L. (2022). *Läraratankar – Kan programmering bidra till lärande i matematik?* Nämnaren 2022:1, s 7–10.

Kilhamn, C., Bråting, K. & Rolandsson, L. (2021). *Programmering i matematik?* Nämnaren, 2021:4, 37–42.

Kilhamn, C., Bråting, K., Helenius, O. & Mason, J. (2022). Variables in early algebra – exploring didactic potentials in programming activities. *ZDM – Mathematics Education*, 54(6), 1273–1288.

Kilhamn, C., Bråting, K. & Rolandsson, L. (2021). Teachers' arguments for including programming in mathematics education. I: G. A. Nortvedt, et al. (red), *Bringing Nordic mathematics education into the future. Preceedings of NORMA 20* (s. 169–176). SMDF.

Raptopoulou, A. (2021). *Politics of contemporary education policy – the case of programming in the Swedish curriculum.* Doktorsavhandling, Stockholms universitet.