Mika Forss

Programmering som verktyg för problemlösning

För att stimulera till fler och bättre examensarbeten med inriktning mot lärande och undervisning i matematik har NCM instiftat ett stipendium i Göran Emanuelssons namn. 2019 års mottagare av stipendiet har analyserat läromedel för gymnasieskolan för att se i vilken utsträckning de ger elever möjlighet att lära sig använda programmering som verktyg för problemlösning.

När två år återstod av min lärarutbildning kom beslutet att programmering skulle skrivas in som en del i grund- och gymnasieskolans matematik. Programmeringens roll blev därför en fråga som naturligt kom upp i samtal med lärare under praktiken och med andra lärarstudenter. Min upplevelse var att många kände en stor osäkerhet om vad programmering i matematiken egentligen skulle innebära. Vilken typ av uppgifter ska vi arbeta med och vad är det egentligen eleverna ska lära sig?

Dessa frågor blev ingången till mitt examensarbete. Nya läromedel, eller komplement till befintliga, hade då börjats släppas utifrån de nya skrivningarna i läroplanen. Med osäkra lärare torde dessa läromedel kunna få en framträdande roll i att forma vad programmering i matematik innebär. Det blir därför viktigt att granska dem kritiskt. Det finns självfallet skäl till en sådan analys på alla skolans stadier, men jag har riktat in mig på programmering i gymnasiematematiken.

För att kunna göra en analys var det viktigt att ha en tydlig grund att utgå ifrån. Den första delen av denna grund ges av styrdokumentens beskrivning av programmering. På gymnasiet återfinns programmering i matematikkurserna Ma1c, Ma2c, Ma3b, Ma3c, Ma4, Ma5 och Ma Specialisering, där formuleringen som rör programmering är identisk för alla kurser och lyder:

> *Strategier för matematisk problemlösning inklusive modellering av olika situationer, såväl med som utan digitala verktyg och programmering.*

Programmeringen beskrivs alltså inom gymnasieskolans matematik som en strategi för problemlösning. Det tolkar jag som att programmeringen framförallt inte ska behandlas i sig självt utan som ett verktyg för eleverna att använda i problemlösning.

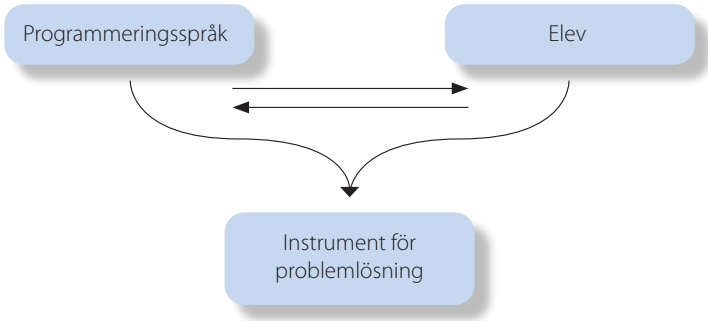
Den andra utgångspunkten för analysen är ett teoretiskt ramverk som beskriver hur vi lär oss att använda verktyg. Jag kommer inte gå in i detalj på alla delar av ramverket här, men ger en kort beskrivning av grundtanken i det som kallas en *instrumentell ansats*. Den som vill läsa mer om detta hänvisas till examensarbetet eller de övriga läsanvisningarna.

Att bygga ett instrument

Den instrumentella ansatsen bygger på distinktionen mellan en artefakt och ett instrument. En artefakt är ett objekt i sig självt, fysiskt eller abstrakt, och ett instrument är någonting mer, något som går att använda för att lösa ett visst problem. Ett exempel kan vara på sin plats här:

> Säg att du vill använda en hammare för att slå ned en spik. Hammaren i sig är då artefakten. Men det räcker inte med bara en hammare. Du måste också ha kunskap om hur man använder en hammare, exempelvis i vilken ände man ska hålla och vad som ska träffa spiken. Först med allt detta kan du använda hammaren för att slå ner din spik.

Allt detta tillsammans, hammaren och kunskapen som gör att du kan använda den, kallas för ett instrument. Att bygga ett instrument utifrån en artefakt lyfts i den här teorin fram som en process i sig. Det är inget som sker automatiskt, utan kan ta tid och kräva mycket arbete. Denna process kallas för en *instrumentell genesis*.



En schematisk bild över den instrumentella genesis som har studerats.

Ska eleverna kunna använda programmering som verktyg för problemlösning i matematik behöver de genomgå en sådan process. De måste bygga ett instrument av programmeringsspråket, lära sig hur det fungerar och strategier för hur det kan användas i problemlösning. Det var den här processen jag ville undersöka genom att ta reda på hur läromedlen stödjer eleverna i deras instrumentella genesis.

Analys av läromedel

Fyra olika material valdes ut för analys. Två gavs ut som komplement till tidigare utgivna läromedel: *Origo* och *Exponent*. Två var uppdaterade och reviderade utgåvor: *Numerus* och *Matematik 5000+*. För den som själv vill titta på materialen eller använda i sin undervisning finns delarna som rör programmering från Origo, Exponent och Matematik 5000+ tillgängligt på respektive förlags webbplats. För avgränsningens skull har jag bara tittat på de material som hör till kursen Ma1c, och bara de delar av respektive läromedel som berör programmering.

Materialen analyserades sen på två nivåer. Först mer allmänt om vilka programmeringsspråk som används, vilken omfattning materialen har, vilka matematiska områden som berörs och vilken typ av uppgifter som finns. Uppgifterna klassificerades utifrån om de bara var av teknisk karaktär eller inte, och om de var problemlösningsuppgifter. Med teknisk karaktär menas

uppgifter som fokuserar programmeringen i sig men som saknar egentligt matematiskt innehåll. Därefter gjordes också en mer ingående analys för att lyfta fram hur och till vilken grad de olika läromedlen ger stöd till elevernas instrumentella genesis, det vill säga att de själva ska kunna tillägna sig programmeringen som ett verktyg för problemlösning.

Exempel på uppgifter

Uppgifterna av teknisk karaktär går ut på att testa att använda en given programkod eller göra mindre förändringar i dem. Utmaningen i uppgiften ligger helt i tekniska aspekter och inte i något matematiskt innehåll.

En uppgift av teknisk karaktär ur Matematik 5000+.

1. Skriv programmet i exemplet. Kör det och lägg märke till antalet pythagoreiska tripplar programmet hittar. Leta efter dubbletter.

2. Ändra programmet i uppgift 1 så att det inte presenterar några dubbletter.

I problemlösningsuppgifterna krävs förutom det tekniska aspekterna också en förståelse för det matematiska innehållet. Lösningen finns inte heller inom direkt räckhåll för eleven genom tidigare exempel. Uppgiften kräver därför att eleven tänker kreativt och arbetar sig fram till en lösning.

En problemlösningsuppgift ur Exponent (uppgift 105).

Du behöver låna pengar och du vill betala av skulden en gång i månaden med ett fast belopp (ett så kallat annuitetslån). Skriv ett program som låter dig mata in lånebelopp, årsräntesats och inbetalat belopp per månad och som beräknar hur lång tid det tar att bli skuldfri.

Studiens resultat

Vi kan börja med att se många likheter mellan läromedlen. Alla har valt *Python* som programmeringsspråk och alla behandlar programmeringen separat, i extramaterial eller i tydligt avgränsade delar av kapitlen. De olika läromedlen har också en motsvarande omfattning på de delar av materialet som berör programmering. Viss variation finns i de matematiska områden som tas upp, men slumphändelser och talteori är vanliga.

Matematik 5000+ och Numerus – tekniska och bakåtsyftande

Matematik 5000+ innehåller en hög andel tekniska uppgifter, uppgifter som saknar ett egentligt matematiskt innehåll. Problemlösningsuppgifter saknas. Uppgifterna i materialet är istället ofta bakåtsyftande, vilket innebär att de går ut på att testköra en kod som är given eller göra mindre förändringar. Materialet saknar också förklaringar av kommandon. De mer tekniska aspekterna berörs istället genom exempelkod.

I Numerus finns också en hög andel tekniska uppgifter och problemlösning saknas. På samma sätt som i Matematik 5000+ är uppgifterna bakåtsyftande, men här behandlas de tekniska aspekterna mycket mer utförligt. Begrepp och kommandon förklaras här direkt för eleven.

Exponent och Origo – fokus på problemlösning

Materialet i Exponent består av några exempeluppgifter med lösningar och uppgifter för eleverna att arbeta med. En betydande del av uppgifterna är problemlösningsuppgifter och eleverna ges utrymme att angripa problem på egen hand. I samband med exempellösningar presenteras en uttalad strategi för problemlösning med programmering. Stödet för programmeringens olika tekniska aspekter är dock knappt. I exempeluppgifterna får eleverna se kod med några kortare kommentarer men förklaringar av olika kommandon saknas.

Origos material består istället av aktiviteter tänkta att genomföras i grupp. Till aktiviteterna finns också lärarhandledning med lösningsförslag men ingen förklarande text eller exempel på programmering finns att utgå ifrån för eleverna. Alla aktiviteter innehåller delar av problemlösning, men problemlösningen bryts också ner i mindre steg i aktiviteterna för att guida eleverna vidare. Det här gör att eleverna får stöd i sitt arbete, men också att de inte får möta problemlösningen på egen hand.

Slutsatser

Vi kan se att de olika läromedlen skiljer sig i vilka aspekter som fokuseras. I Origo och Exponent är det problemlösning med programmering som är i fokus. I kontrast till detta är det i Numerus och Matematik 5000+ istället de tekniska aspekterna som behandlas. Vidare ser vi att det finns skillnader i hur man väljer att behandla de olika aspekterna. I Origo ges ett tydligare

stöd i arbetet med problemlösningen medan Exponent ger eleverna ett större utrymme att angripa problemen på egen hand så att eleverna får möta problemlösningen mer som en helhet. Eftersom båda aspekterna är viktiga kanske materialen också kan komplettera varandra. Mellan Numerus och Matematik 5000+ ser jag en mer avgörande skillnad i hur de tekniska aspekterna behandlas. Numerus gör det mycket mer explicit med förklaringar av kommandon och begrepp. Sett till behandlingen av tekniska aspekter kan Numerus därför vara att föredra som läromedel.

När det gäller skillnaden i fokus mellan problemlösning och tekniska aspekter är det viktigt att komma ihåg att båda delar behövs. Eleverna behöver möta och lära sig hantera de tekniska aspekterna och också ta steget vidare in i problemlösning med programmeringen. Utifrån detta saknar alltså alla de studerade läromedlen viktiga bitar. Hur mycket som behöver behandlas av grundläggande tekniska aspekter beror dock också på förkunskaper hos eleverna, något som är svårt att veta på förhand idag medan programmeringen samtidigt också införs i de lägre stadierna. Det kan dock vara lätt att glömma bort problemlösningen om man som lärare utgår från ett läromedel som bara berör de mer tekniska aspekterna. Eftersom det var just problemlösningen som programmeringen skulle syfta till kan materialen från Origo och Exponent därför vara att föredra.

Oavsett vilket läromedel som används är det tydligt att vi som lärare behöver förhålla oss självständiga gentemot de material som används i undervisningen. Vill vi uppfylla styrdokumentens beskrivning kring programmering och ge eleverna stöd att själva kunna tillägna sig programmering som ett verktyg i problemlösningen behöver vi arbeta både med de tekniska aspekterna och med problemlösning. Utifrån den teoretiska ram som jag har använt vill jag också lyfta fram vikten av att betrakta detta som en process i sig, och förmodligen en utdragen och tidskrävande sådan. Om eleverna ska kunna använda programmering måste de också lära sig att hantera detta nya verktyg. Det kommer att kräva tid och arbete, för såväl elever som lärare.

LITTERATUR

Drijvers, P., Doorman, M., Boon, P., Reed, H. & Gravemeijer, K. (2010). The teacher and the tool: Instrumental orchestrations in the technology-rich mathematics classroom. *Educational Studies in Mathematics*, 75(2), 213–234.

Forss, Mika. (2019). *Att bygga ett instrument – En instrumentell ansats till programmering som ett verktyg för problemlösning i läromedel för matematik*. Länk till arbetet: gupea.ub.gu.se/handle/2077/62070

Guin, D. & Trouche, L. (1998). The complex process of converting tools into mathematical instruments: The case of calculators. *International Journal of Computers for Mathematical Learning*, 3(3), 195–227.