Kommunikation i olika programmeringsmiljöer

Constanta Olteanu och Lucian Olteanu, Linnéuniversitetet

En av förmågorna som eleverna ska utveckla med hjälp av undervisningen i matematik är förmågan att kommunicera med och om matematik, dvs. att utbyta information med andra om matematiska idéer och tankegångar, muntligt, skriftligt och med hjälp av olika uttrycksformer (Skolverket, 2017). För att utveckla elevernas förmåga att kommunicera med och om matematik i programmering ska eleverna få konstruera, beskriva och följa användningen av en algoritm i visuella programmeringsmiljöer.

Symboler i matematik

Symbolerna som används i matematik i de första årskurserna handlar i första hand om siffersymboler (0, 1, 2 ... 9) och symboler för räknesätt (+, -, • och ÷). Det finns även symboler som används för att uttrycka relationer, dvs. ett förhållande mellan ett antal händelser med avseende på en viss egenskap. Det kan finnas två typer av relationer:

- Enkla, där man jämför två händelser. I matematik används symbolerna =, ≠, > och < för att uttrycka enkla relationer.
- Sammansatta, där man jämför fler än två händelser. Relationer såsom ”att ligga emellan” inbegriper tre händelser. Exempelvis, ”2 ligger mellan 1 och 3” kan skrivas som ”1 < 2 och 2 < 3”.

Instruktioner i vilka matematiska symboler används kan vara både muntliga och skriftliga. Många instruktioner finns i läroböcker. För att eleverna ska kunna lösa en uppgift i en lärobok eller ett prov måste eleverna förstå vad de ska göra. Det är vanligt att det blir fel bara för att eleverna missar något i uppgiftens instruktion.

Siffersymboler och symboler för olika räknesätt används i programmering på samma sätt som i algebra och aritmetik. Detta innebär att eleverna kan använda sina kunskaper inom dessa områden för att ge instruktioner som en dator ska genomföra, det vill säga att programmera. Ett undantag i flera programspråk är likhetstecknet. Det som främst skiljer användningen av likhetstecknet i algebra och i programmering presenterades i texten ”Programmering och programmeringsprocessen”.

Symboler i programmering

Det finns studier som visar att elever kan utveckla sin resonemangs- och kommunikationsförmåga med hjälp av programmering (Calder, 2010; Clements, 2002). Detta sker genom att elever diskuterar och utforskar komplicerade begrepp, testar sina program på en kamrat eller lärare när de undersöker hur ett objekt ska förflytta sig, kommunicerar med andra uttrycksformer samt överför sina nyvunna kunskaper i andra

situationer. Att låta eleverna skapa instruktioner som någon annan ska utföra visar på vikten av tydlighet.

Genom stegvisa instruktioner och användande av olika symboler förklarar eleven för datorn eller för en klasskamrat vad den ska göra och i vilken ordning instruktionerna ska ges så att uppgiften blir korrekt utförd. De vanligaste symbolerna som används för att ge stegvisa instruktioner är pilar och bilder. Man kan rita dessa själv eller ta färdiga bilder från internet.

Att utveckla förmågan att bryta ner problem i mindre delar och ge entydiga instruktioner är centralt för att en dator ska kunna tolka och köra ett program. Det finns inte någon möjlighet att komplettera med förklaringar för en dator, en dator gör som du säger och inte som du tänker.

När man programmerar är det en fördel om man är uppmärksam på instruktioner eller sekvenser som återkommer, så att man slipper skriva samma sak flera gånger. I början spelar det inte så stor roll, men ju mer komplicerade instruktioner man ger, desto viktigare blir det att eleverna ser skillnader och likheter mellan användningen av olika symboler.

Visuella programmeringsmiljöer

I visuella programmeringsmiljöer kan eleverna till exempel ”dra-och-släppa” fördefinierade grafiska element för att sätta samman sina program. Detta kallas i vardagligt tal ofta för blockprogrammering (Skolverket, 2017a). Visuell programmering låter eleverna beskriva processer med hjälp av grafiska illustrationer. Valet av en programmeringsmiljö beror på till exempel lärandemål, elevernas förkunskaper och problemets typ. Oavsett vilken programmeringsmiljö du som lärare väljer att använda i ditt klassrum måste du veta att det finns olika sätt att beskriva en algoritm.

Nedan ges fyra exempel på hur man kan skriva en algoritm som beskriver spelet Hilo - med naturligt språk, blockdiagram, pseudokod och strukturdiagram. Spelet Hilo går till så att datorn väljer ett slumpmässigt tal och spelaren skall gissa vilket tal det är. För varje gissning skall datorn tala om gissningen är rätt, högre eller lägre än rätt svar. Här beskrivs en enkel variant där det enbart tar upp en gissning.

- **Naturligt språk** - eleverna förklarar en algoritm, där variabler ingår, med vanlig text.

  Uppmana användaren att mata in ett tal som ska vara hemligt.

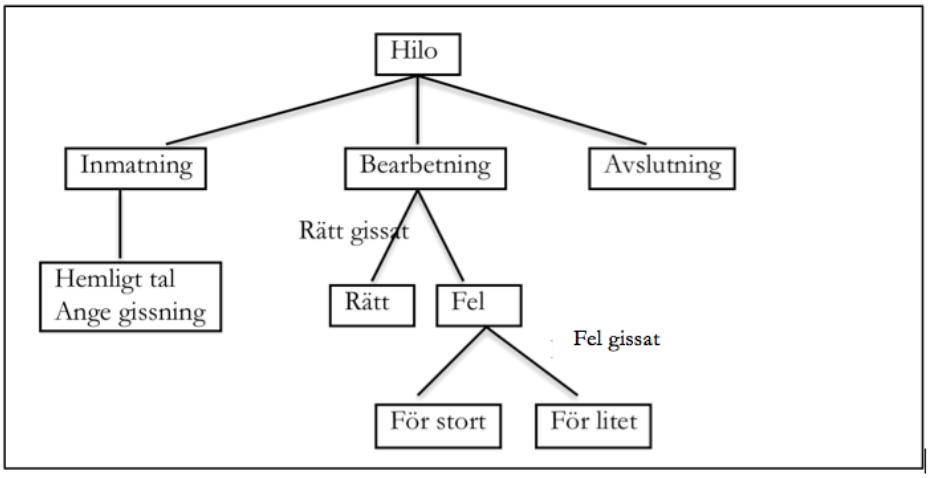
  Uppmana användaren att mata in ett gissning tal.

  Kontrollera om gissat tal är lika med hemligt tal.

  I så fall skriv ut ”Rätt”.

  I annat fall skriv ut ”För stort” eller ”För litet”.

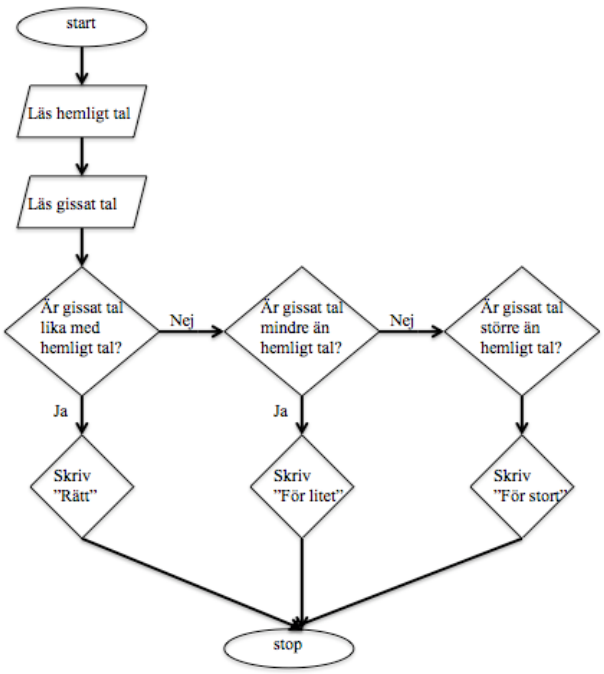
- **Blockdiagram** - eleverna visar en grov struktur av en algoritm i form av ritade "boxar". Varje box kan vara en del av algoritmen.



- **Pseudokod** - man skriver en blandning av programspråk, vanlig text och matematik. Pseudokod används både vid beskrivning av algoritmer och vid arbete med att utveckla dem, men är inte körbar på en dator. En pseudokod ska vara lättläst, tillräckligt detaljerad kort och inte innehålla deklaration av variabler. Exempel:

```
Ange hemligt tal.

Ange gissning.

Om rätt gissat tal, skriv ut ”Rätt”

annars

    Om för stort tal, skriv ut ”För stort”

annars

    Om för litet tal, skriv ut ”För litet”
```

- **Strukturdiagram** - man ritar algoritmen med olika symboler, som visar när och hur saker skall ske i programmet.



En elips används för att markera programstart och slut. En parallellogram anger input och en romb betyder val eller beslut och kan på olika sätt symbolisera selektion eller en del av en loop

Blockprogrammering

Under de senaste åren har blockbaserade programmeringsmiljöer blivit populära. Denna våg av popularitet har blivit uppmuntrad av Scratch, men det finns flera andra liknande miljöer som till exempel Blockly och Snap. Det finns flera likheter mellan dessa visuella programmeringsmiljöer. De har en färdig uppsättning instruktioner i form av block som man kan pussla ihop för att bygga sina program. De är dessutom utvecklade för att användas av nybörjare och ställer därför oftast inga stora krav på varken tekniska förkunskaper eller tidigare erfarenheter av programmering. Blockbaserade språk minimerar också möjligheten till syntaxfel och behovet av att memorera namn på olika instruktioner eftersom de finns färdigt i form av block. På så sätt kan man fokusera på att formulera algoritmen, det vill säga lösa problemet, i stället för att fokusera på att skriva allt exakt rätt enligt språkets grammatik.

En anledning till att blockprogrammering har blivit så populärt de senaste åren är att eleverna kan se alla bitar i blockbaserade miljöer. På så sätt är det lättare för dem att upptäcka vilka instruktioner och operationer som är möjliga att genomföra med hjälp av en dator. En annan anledning är att färger hjälper eleverna att gruppera operationer logiskt

medan blockens form för det mesta gör det möjligt att avgöra om olika instruktioner och operationer fungerar tillsammans.

Flera studier (t.ex. Weintrop & Wilensky, 2015) visar att det finns tre stora fördelar med användningen av blockprogrammering. Dessa är:

1) Enkel åtkomst till kraftfulla visualiseringar.
Exempelvis kan eleverna enkelt göra en sprajt (eng. sprite), det vill säga ett objekt som kan kontrolleras genom programmering och kan till exempel vara en haj eller en sjöstjärna. Med hjälp av instruktioner kan eleverna få dem att ändra riktning, utseende och interagera med andra sprajtar.

2) Lättanvända kontrollstrukturer, såsom händelsehantering.
Eleverna kan exempelvis få flera sprajtar att flytta sig samtidigt och få dem att skicka meddelanden till varandra. Det kan handla om två apor kommunicerar med varandra. Alternativ (villkorssatser/if-satser) och repetition (slingor/loopar) är lätt att förverkliga. Eleverna kan också enkelt ge olika instruktioner, som till exempel att spela upp ett ljud eller genomföra beräkningar. I textbaserade programspråk räcker det sällan med enskilda instruktioner för att åstadkomma denna typ av händelser.

3) Möjligheten att bygga riktiga program och dela dessa med andra elever på samma skola eller på andra skolor.

Blockbaserade programmeringsmiljöer är utformade för att säkerställa att nybörjarprogrammerare får positiva erfarenheter av att kommunicera med en dator genom att få visuella signaler om hur, var och varför olika instruktioner används för att skapa en algoritm.

Textbaserad programmering

I läroplanen för åk 7-9 talar man om olika programmeringsmiljöer. Detta kan förutom visuella programmeringsmiljöer också handla om textbaserad programmering. Ett textbaserat programmeringsspråk används för att utveckla mer komplicerade program. Det finns program, som till exempel Blockly, som gör det möjligt att växla mellan blockprogrammering och textprogrammering. För att detta ska fungera krävs att man har helt entydiga regler för hur olika saker ska skrivas och att skrivsättet och beteckningssystemet är mycket väl definierat. Programspråk karakteriseras av att ha både syntax (språkregler) och semantik (betydelser). Syntaxen beskriver språkets form, medan betydelsen, effekten eller innebörden av exempelvis olika instruktioner, som förekommer i skapandet av en algoritm, beskrivs av språkets semantik. Det som skrivs för att få en dator att utföra något kallas för kodning. Ett flertal forskare (t.ex. Crow, 2014; Nettleford, 2013) påpekar att programmering inte enbart betyder "kodning" och att betrakta programmering som enbart kodning kan ge en missvisande bild för användningen av programmering i en undervisningssituation.

Det finns många olika programspråk, men det är mindre viktigt vilket programspråk du som lärare väljer att jobba med, förutsatt att det ger eleverna möjlighet att utveckla en generell förståelse av programmeringsprinciper och dess användning (Skolverket 2017a).

Kommunikation i programmering

Ett flertal studier har visat att när eleverna arbetar i par, *parprogrammering*, skapas en miljö som bidrar till aktivt lärande och social interaktion (t.ex. Nagappan, 2003). Detta arbetssätt gör det möjligt för eleverna att utveckla sin förmåga att arbeta kollaborativt vilket innebär att man lär tillsammans och att man kan utnyttja varandras kunskaper och kompetenser. Genom att arbeta kollaborativt utvecklar eleverna sin kommunikationsförmåga och förmågan att samarbeta. De får också träna på problemlösning och kritiskt tänkandet genom att inte enbart lära sig av varandra utan även med varandra.

När eleverna arbetar i par ger de instruktioner till varandra. Den ena är sändare eller programmerare och den andra är mottagare eller dator. Eftersom mottagaren genomför olika handlingar, muntliga, skriftliga eller visuella, utifrån sändarens instruktioner är det viktigt att sändaren reflekterar över vilka instruktioner som ska ges och i vilken ordning. Mottagaren avkodar instruktionernas betydelse och innebörd. På grund av denna avkodningsprocess finns det alltid en risk för att det ska uppstå missförstånd som gör att en eller flera svårigheter blir synliga. För att minska dessa missförstånd är det viktigt att sändaren får återkoppling, för att få veta om mottagaren verkligen förstått instruktionen. Om missförstånd uppstått måste sändaren alltid åtgärda dessa genom att korrigera de instruktioner som ges.

När eleverna kommunicerar utför de kodning, tolkning och avkodning av en algoritm. Kommunikation är ett viktigt element för att effektivisera skapandet och användningen av en algoritm. Detta leder i sin tur till en lösning av ett problem på ett enklare och tydligare sätt.

Transfer

Att överföra och använda kunskaper från ett område till ett annat kallas transfer (Kjällander, m.fl., 2015). Transfer mellan programmering och algebra kan till exempel uppmärksammas när eleverna ger instruktioner till varandra för att utföra olika operationer med tal eller för att tolka användningen av tal och symboler. Om eleverna ska kunna ge entydiga instruktioner för att finna det okända talet i uttrycket $5x - 3 = 2x - 3$ måste eleverna använda kunskap från aritmetik och algebra. Användningen av blockprogrammering möjliggör för eleverna att behålla fokus på användningen av den algebraiska kunskapen i relation till programmering och tvärtom. Detta innebär att eleverna exempelvis kan använda algebraiska uttryck för att kunna programmera samtidigt som de kan tolka vissa steg som används i en blockprogrammering med hjälp av algebraiska uttryck. Dessutom möjliggör användningen av blockprogrammering att kommunikationen är begriplig och enkel eftersom visuell programmering låter eleverna beskriva processer med hjälp av illustrationer.

Det finns möjligheter till lärande i matematik när man arbetar med programmering i undervisningen. Dock har läraren en central roll genom att påvisa och lyfta in matematiken i undervisningen, då den annars kan bli osynlig för eleven. I exemplet ovan, där elever testar sina program på en kamrat eller lärare, kan läraren väva in matematiska begrepp genom att personen som blir programmerad ska förflytta sig ett visst antal grader eller i en viss vinkel.

Förslag på aktiviteter

Eleverna genomför samtliga aktiviteter i par.

Aktivitet 1 – olika programmeringsmiljöer

Utrusning: Dator.

Förberedelser: Skriv ut och kopiera stenciler med beskrivningarna som handlar om spelet Hilo (ligger under rubriken ”Visuella programmeringsmiljöer”). Välj en programmeringsmiljö. Se till att eleverna har samma program installerat på sina datorer.

Syfte: Syftet med denna aktivitet är att eleverna ska se sambandet mellan olika sätt att beskriva en algoritm.

Matematiskt innehåll: jämförelse av tal

Genomförande: Låt varje par formulera en beskrivning av samma algoritm genom att använda blockprogrammering eller ett textbaserat språk såsom JavaScript eller Pyth§on. Låt även eleverna jämföra instruktionerna som används i olika programmeringsmiljöer och diskutera fördelar och nackdelar med varje miljö i olika situationer.

Aktivitet 2 - blockprogrammering och programmeringsspråk

Utrusning: Dator.

Förberedelser: Skriv ut och kopiera stenciler med problemet som föreslås i denna aktivitet. Välj en programmeringsmiljö. Se till att eleverna har samma program installerat på sina datorer.

Syfte: Syftet med denna aktivitet är att låta eleverna jämföra blockprogrammering med textbaserad programmering.

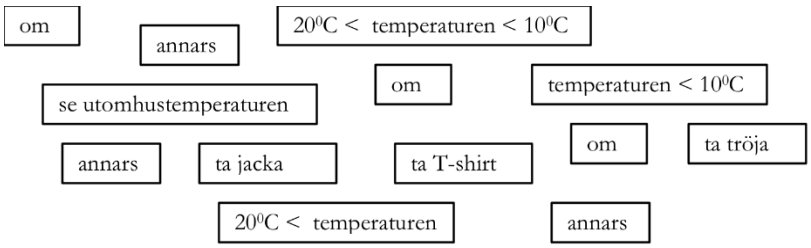
Matematiskt innehåll: olikheter

Genomförande: Låt eleverna arbeta i par för att skapa ett blockbaserat program genom att använda givna kodblock.

Ge eleverna följande problem:

> Per väljer kläder beroende på utomhustemperaturen:
>
> - Han tar en T-shirt om det är varmare än 20^0C ute.
> - Han tar en tröja om temperaturen är mellan 10^0C och 20^0C.
> - Han tar en jacka om temperaturen är under 10^0C.

Skapa en algoritm som hjälper Per att välja hans kläder genom att pussla ihop följande instruktionsdelar. Skapa därefter ett blockbaserat och ett textbaserat program.



Låt eleverna jämföra instruktionerna som används i ett blockbaserat och ett textbaserat program och diskutera fördelar och nackdelar med varje miljö.

Aktivitet 3 – Att tolka det resultat algoritmen ger

Utrusning: Dator.

Förberedelser: Skriv ut och kopiera instruktionerna och mönster som föreslås i denna aktivitet eller skapa andra instruktioner och mönster. Dela ut instruktionerna till varje par. Välj en programmeringsmiljö. Se till att eleverna har samma program installerat på sina datorer.

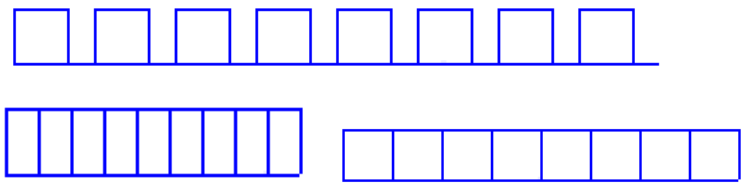
Syfte: Att ge eleverna möjlighet att tolka resultatet som en algoritm ger och välja rätt svar bland olika alternativa svar.

Matematiskt innehåll: förflyttning, vinklar, mönster

Genomförande: Ge eleverna olika mönster att välja mellan och fråga vilka av dessa mönster som motsvarar algoritmen nedan. Eleverna resonerar med varandra om vilket mönster som motsvarar algoritmen nedan.

- Gå 20 steg.
- Vänd 90^0 åt vänster.
- Gå 20 steg.
- Vänd 90^0 åt vänster.
- Gå 20 steg.
- Vänd 90^0 åt vänster.
- Gå 40 steg.
- Repetera 8 gånger

Exempel på mönster:



Därefter kan den ena eleven i varje par vara programmerare och skapa ett blockprogram för att visualisera mönstret. Den andra eleven granskar varje instruktion när algoritmen skrivs. När programmet körs har eleverna möjlighet att direkt få återkoppling på sina resonemang och tolkningar. Eleverna ska sedan skapa motsvarande program i ett textbaserat programspråk. Låt eleverna studera vad som händer om man vänder 90^0 eller 120^0 åt höger. Låt därefter eleverna diskutera likheter/skillnader/fördelar/nackdelar med olika programmeringsmiljöer. Sammanfatta för hela klassen vad eleverna har kommit fram till.

Aktivitet 4 - algoritmer

Utrusning: Dator.

Förberedelser: Skriv ut och kopiera mönster och kodblocken som föreslås i denna aktivitet eller skapa andra mönster och kodblock. Dela ut mönster och kodblock till varje par. Välj en programmeringsmiljö och ett textbaserat programspråk. Se till att eleverna har samma program installerat på sina datorer.

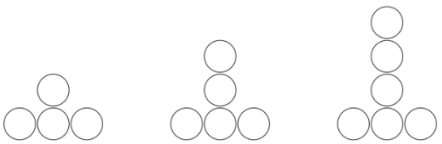
Syfte: Aktiviteterna som föreslås nedan syftar på att låta eleverna skriva egna algoritmer.

Genomförande: Eleverna arbetar i par och presenterar därefter sina algoritmer i storklass. Utvärdera varje algoritm med eleverna och diskutera om det finns potential för att göra algoritmerna bättre, till exempel genom att använda slingor/loopar eller andra typer av

generaliseringar. Låt eleverna skapa program som motsvarar algoritmerna i en programmeringsmiljö och diskutera skillnader och likheter mellan block- och textprogrammering.

1. Skriv en algoritm som visar antal föremål i figur 10. Skriv därefter en formel som visar antal föremål (bollar, stickor) i figur 100.

a)

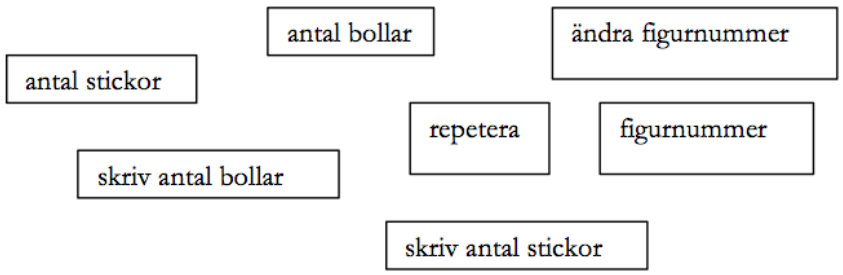


Figur 1 Figur 2 Figur 3

b)

Figur 1 Figur 2 Figur 3

Eleverna skulle kunna använda följande kodblock:



Referenser

Calder, N. (2010). Using Scratch: an integrated problem-solving approach to mathematical thinking. *Australian Primary Mathematics Classroom*, 15(4), 9–14.

Clements, D. H. (2002). Computers in Early Childhood Mathematics. *Contemporary Issues in Early Childhood*, 3(2), 160-181.

Crow, D. (2014). *Why Every Child Should Learn to Code.* http://www.theguardian.com/technology.

Kjällander, S., Åkerfeldt, A., & Petersen, P. (2015). *Översikt avseende forskning och erfarenheter kring programmering i förskola och grundskola.* Skolverket.

Nagappan, N., Williams, L., Ferzli, M., Yang, K., Wiebe, E., Miller, C. & Balik, S. (2003). *Improving the CS1 Experience with Pair Programming, in ACM Special Interest Group Computer Science Education (SIGCSE),* s. 359 - 362.

Nettleford, W. (2013). *Primary School Children Learn to Write Computer Code*. Available: http://www.bbc.co.uk/ news/uk-england-london-23261504.

Shu, N.C. (1988). *Visual Programming.* Van Nostrand Reinhold, New York.

Skolverket. (2017, reviderad). *Läroplan för grundskolan, förskoleklassen och fritidshemmet*. www.wolterskluwer.se/offentligapublikationer

Skolverket. (2017a, reviderad). *Kommentarmaterial till kursplanen i matematik.* https://www.skolverket.se/laroplaner-amnen-och-kurser/grundskoleutbildning/grundskola/matematik

Weintrop, D., & Wilensky, U. (2015). To block or not to block, that is the question: students' perceptions of blocks-based programming. *Proceedings of the 14th International Conference on Interaction Design and Children*, ACM, pp. 199–208.