Stenseth, Kaufmann, Forsström

Programmering og matematikk

Innledning

Programmering kan betraktes som en grunnleggende ferdighet for å delta effektivt i den digitale verden, og det er en økende interesse for å introdusere programmering som skolefag (Grover og Pea, 2013; Balanskat og Engelhardt, 2015). Programmering kan defineres som prosessen knyttet til utvikling og implementering av instruksjoner for dataprogrammer slik at datamaskinen kan utføre spesifikke oppgaver, løse problemer og støtte menneskelige interaksjoner. Derfor krever programmering generelt at de som programmerer, har kunnskap om programmeringsspråk, ekspertise innen fag relatert til utvikling av spesialiserte algoritmer og logikk og evne til å analysere, forstå og løse problemer ved å verifisere algoritmiske krav og vurdere korrekthet og implementering (ofte referert til som koding) av algoritmen i et bestemt programmeringsspråk. Fordi disse prosessene ofte knyttes til matematisk tenkning og algoritmisk tenkning (Grover og Pea, 2013), er det blitt en viktig ferdighet for det digitale samfunn og framtidens kompetanse innen problemløsning, kreativitet og logisk tenkning. Det er derfor ikke en debatt om programmering skal inn i skolen, men på hvilken måte det kommer inn. I økende grad har land i Europa integrert programmering som del av andre fag (Balanskat og Engelhardt, 2015), også i matematikk. I Norge skal vi som kjent integrere programmering i matematikkfaget, noe som har ført til stor debatt. Vi ønsker å fokusere på hvilke muligheter det vil gi matematikkfaget. Vi mener det er mer fruktbart å fokusere på ulike metoder ved programmeringsfaget som kan bidra inn mot matematikkfaget, i stedet for de mer «tekniske» sidene ved programmering, som at elevene «*skal lage algoritmar med bruk av variablar, vilkår og lykkjer og programmere desse*» (kompetansemål for 5. trinn i forslag til ny læreplan) og «*bruke variablar, lykkjer, vilkår og funksjonar i programmering til* å utforske geometriske *figurar og mønster*» (kompetansemål for 6. trinn).[1] Derfor vil vi argumentere for hvordan programmeringsfaget og matematikkfaget kan for forsterke hverandre. Det som er interessant og utfordrende, er om metoder i programmeringsfaget kan bidra med tenkning og problemløs-

Børre Stenseth
Pensjonist
borre.stenseth@gmail.com

Odd Tore Kaufmann
Høgskolen i Østfold
odd.t.kaufmann@hiof.no

Sanna Erika Forsstöm
Høgskolen i Østfold
sanna.forsstrom@hiof.no

Stenseth, B., Kaufmann, O. T. & Forsström, S. E. (2019). Programmering og matematikk. *Tangenten – tidsskrift for matematikkundervisning, 30*(2), 7–12.

www.caspar.no/tangenten

ningsmetodikk som styrker matematikken. Vi vil peke på noen elementer og ut fra erfaringer drøfte noen utfordringer.

Algoritmisk tenkning

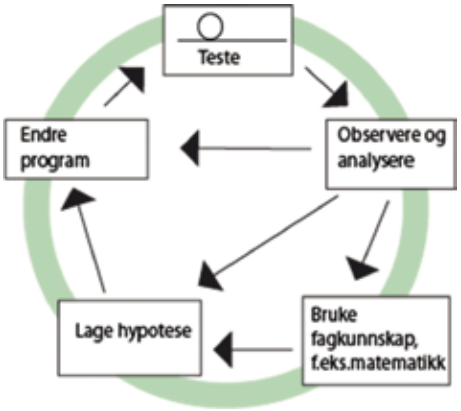
Det som nevnes i mange sammenhenger, også i fagplanene, er algoritmisk tenkning. Definisjonen av dette og vektlegging av forskjellige komponenter varierer en del. Hvis vi skal koke det ned til noen enkle begreper, er det vel å utvikle problemløsningsmetoder ved å dekomponere, abstrahere og skaffe seg en oversikt over problemet og mulige løsningsstrategier. I forslag til ny læreplan er algoritmisk tenkning beskrevet slik:

> Algoritmisk tenking er viktig i prosessen med å utvikle strategiar og framgangsmåtar og inneber å kunne bryte ned eit problem i delproblem som kan løysast systematisk. Viktig innhald vil vere å kunne stille matematiske spørsmål og formulere matematiske problemstillingar, identifisere problem, vere uthaldande, utvikle og kunne velje effektive problemløysingsstrategiar og utforske og løyse problem ved hjelp av programmering.

I sin generelle form brukes begrepet som en metode for å finne og beskrive metoder både for mennesker og maskiner. Slik sett kan vi si at en strikkeoppskrift er et resultat av en algoritmisk tilnærming. I den konteksten vi bruker begrepet, vil en algoritmisk angrepsvinkel fokusere definisjon av variabler og bruk av sekvenser, betingelser og løkker. Altså begreper vi må mestre for å få programmer til å fungere. Vi kan godt legge til funksjoner, moduler og gjerne objekter.

Feilretting

Feilretting, eller korreksjon, er en viktig del av arbeidet med programutvikling. Her skiller programmering seg litt fra strikkeoppskrifter, siden de er vesentlig lettere å rette, i alle fall rent teknisk. Ingen programmer er «riktige» første gang de skrives og testes. Det kan være syntaksfeil eller kjøretidsfeil, eller det kan være at programmet gir uønskede svar. Dette er underkommunisert i programmeringsundervisning og bidrar ofte til at programmering oppfattes som vanskelig og rigid i starten. Feilretting kan imidlertid snus til en positiv prosess. Det underbygger resonnementer som innebærer hypoteseformulering og hypotesetesting. Feilretting innebærer en iterativ innfallsvinkel til problemløsning. Det innebærer at arbeidet foregår i en syklus. Vi kan illustrere dette som i figur 1.



Figur 1

Basert på våre erfaringer kan vi beskrive tre ulike typer iterasjoner eller feilrettingsforsøk:

Den intuitive. Her kjøres programmet, det gjøres en observasjon og kanskje en svært enkel analyse. Programmet modifiseres litt tilfeldig og kjøres igjen.

Den kodefokuserte. Her tolkes effekten av variabler og av kodesekvenser. For eksempel vil en variabel som beskriver bredden på et rektangel, kunne endres for å se om det passer med andre deler av den figuren vi tegner.

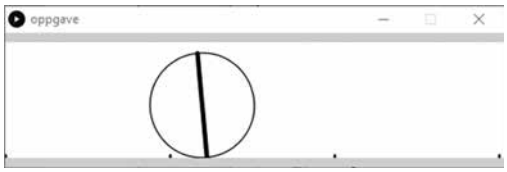
Den teoretisk funderte. Her brukes matematikken, geometrien, til å analysere hva som skjer i et eksperiment og til å formulere en hypotese. Det er hit vi gjerne vil at elevene kommer. Det betyr at de får visualisert sine matematiske hypoteser.

Stenseth, B., Kaufmann, O. T. & Forsström, S. E. (2019). Programmering og matematikk. *Tangenten – tidsskrift for matematikkundervisning, 30*(2), 7–12.

www.caspar.no/tangenten

Erfaringer fra et eksperiment

Vi kjørte et eksperiment med geometri på 9. trinn for å undersøke hvordan problemløsning via feilretting fungerte. Oppgaven gikk i korthet ut på å korrigere et program som tegnet et hjul som beveget seg bortover en flate. Hjulets hastighet skulle endres slik at rullingen ble «naturlig», altså uten skrensing eller spoling, uavhengig av hjulets radius. Elevene fikk utdelt et program som fungerte, men som altså rullet med feil hastighet. Programmet elevene fikk



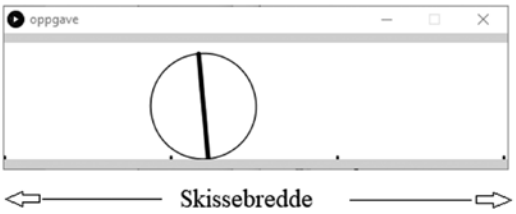
Figur 2

utdelt ligger på Tangentens nettside (caspar.no/tangenten/2019/programmering.pdf).

Tre elever samarbeidet om problemet foran samme PC, og hele arbeidssekvensen ble videofilmet og analysert i detalj. En uke før eksperimentet hadde elevene fått en innføring i Processing (https://processing.org/), som var det utviklingsverktøyet som ble brukt. Elevene arbeidet sammen i omtrent en time med oppgaven. Elevene behersket verktøyet og hadde få eller ingen problemer med å endre koden slik at den forble kjørbar. Analysen av den iterasjonen elevene gjorde, var at de startet med det vi ovenfor har kalt intuitive eksperimenter. Nedenfor følger en kort beskrivelse av det de gjorde:

Det første elevene gjør, er å kjøre programmet for å se hvordan koden fungerer. De finner ut at de må endre rotasjonshastigheten, og har en meningsutveksling om hvordan det kan gjøres. De endrer variabelen *re* som står for rotasjonsendring. På dette tidspunktet har elevene ikke skaffet seg noe holdepunkt for hvor mye de skal endre, og de forsøker med prøving og feiling:

Petter	Vent a (leser fra koden på skjermen). Det er den her, *float re*. Er det ikke bare fire kanskje? (Endrer verdien på *re* fra 8 til 4 og kjører programmet)
Sander	Da halverer du rotasjonen?
Martin	Åh.
Petter	Nesten.
Martin	Nesten. Fem, prøv, fem.
Petter	Ja. Skal vi se. Jeg tror kanskje ikke det er sånn man skal gjøre det. Nei.



Figur 3

Etter at de har forsøkt å endre på verdien *re* noen ganger, fokuserer de deretter på skissebredden, *width* (figur 3).

En hypotese som dukker opp, er å ta skissebredden og dele på antall rotasjoner som er merket av på skissen. De bruker kalkulatoren og finner ut at 472/5 = 94,4. Resonnementet svikter når de kommer på at *re* (variablene for rotasjonsendring) er grader, ikke antall rotasjoner. Dette fører til en analyse av kode og variabler. De slår nå fast at variabelen *pe* (posisjonsendring) er piksler og variablene *re* (rotasjonendring) er grader:

Petter	Vi kan ikke bare teste veien. Firehundreogsyttito delt på fire da.
Martin	Det er fem streker da.
Petter	En to tre fire fem. Det er fem. Firehundreogsyttito delt på fem. [Slår inn 472/5 på kalkulatoren.] Nittifire komma fire.
Martin	Men hva skulle vi trenge nittifire komma fire til?
Petter	For å finne ut hvor langt det er mellom …

Stenseth, B., Kaufmann, O. T. & Forsström, S. E. (2019). Programmering og matematikk. *Tangenten – tidsskrift for matematikkundervisning, 30*(2), 7–12.

Sander	Det er mellomrommet fra pikslene. Hver strek til strek.
Petter	Den roterer fire, fire, er det grader? [Kommenterer *re*, som nå er 4.]
Sander	Det er fire rotasjoner.
Petter	Nei, det er fire grader.

Etter en del forsøk med prøving og feiling begynner de å finlese koden på nytt.

De stiller spørsmål om det er rett å gå videre med 94,4 (skissebredde/antall markeringer på skissen). Gruppa begynner nå å se på rotasjonsendring som en funksjon av hjulradius. De setter deretter rotasjonsendringen (*re*) til: radius/78,6 · 5, der 78,6 = *width*/6. Dette ser ut til å fungere visuelt når de kjører koden. Det skjærer seg med dobling av radius.[2] Det viser seg å være vanskelig å formulere i kode, og de finner ikke en brukbar formulering.

På slutten av leksjonen kommer gruppa inn på omkrets som operativt begrep etter hint fra læreren. Elevene har for det meste fram til nå fokusert på diameter og radius av en sirkel, mens en omdreining av sirkelen tilsvarer sirkelens omkrets. Dette ville ledet gruppa framover, men de har dessverre for liten tid til å fortsette.

Lærer	Det er sentralt at det er en sirkel da. Se her. Hvor langt er det den dreier i løpet av en omdreining?
Petter	Trehundreogseksti grader eller diameteren.
Lærer	Ikke diameter.
Petter	Nei, radius.
Lærer	Nei, det er ikke radius heller. Det er ikke diameter og ikke radius.
Petter	Omkretsen.
Lærer	Omkretsen, ja. Helt riktig. På én omdreining så har den. Dere har snakket om radius hele tiden. Det er omkretsen.
Petter	Radius ganger to ganger pi, det er det det samme som diameter ganger pi. [Prøver nå å sette *re* til radius · 2 · pi.]
Martin	Oi oi oi.

Et problem som bidro til at elevene kom skjevt ut, var at den visuelle responsen sa dem at hjulet rullet riktig, mens det i virkeligheten rullet dobbelt så fort som det skulle. Helt på slutten av eksperimentet, etter ca. 1 times arbeid, innså de dette. Først da ble matematikken eksplisitt inkludert i hypotesene som ble testet. Gruppa samarbeidet sosialt helt utmerket, men arbeidsdelingen og arbeidet med hypoteseformuleringer kunne vært bedre forberedt. En av elevene var på sporet av den misforståelsen som er nevnt ovenfor, men innvendingene hans ble ikke skikkelig analysert i gruppa. Motivasjonen og ønsket om å løse oppgaven var høy under hele økta. Elevene ville helst ikke avslutte oppgaven selv om timen var slutt.

Ut fra erfaringene har vi identifisert noen endringer vi kan gjøre neste gang vi prøver ut noe lignende: Vi kan tydeliggjøre koplingen mellom matematikk og kode bedre i programmet de får som utgangspunkt for arbeidet. Det gjelder variabelnavn, kommentarer og kanskje til og med en eksplisitt angivelse av hvilke variabler som kan/må endres.

Vi kan tydeliggjøre og bevisstgjøre iterasjonssekvensen de går inn i, blant annet ved å si litt om samarbeid og hypoteseformulering.

Dette kan gjøre at elevene raskere kan komme fram til akseptable løsninger, for eksempel ved å fokusere tydeligere på hvor koden skulle endres, slik det er vist nedenfor:

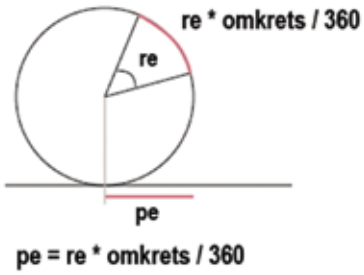
Original kode (utdrag av det utdelte programmet) i figur 5. Det ville kanskje gjort det enklere å komme fram til noe slikt som i figur 6, basert på en geometrisk betraktning som i figur 4.

Noen tanker i etterkant

Det er interessant å se litt mer på utviklingen i programmeringsfaget. Vi har pekt på algoritmisk tenkning, vi har fokusert på feilretting, og vi har analysert en iterativ arbeidsform. Hvis vi ser på utviklingen i informatikkfaget, finner vi at det blir fokusert stadig mer på planlagte,

Stenseth, B., Kaufmann, O. T. & Forsström, S. E. (2019). Programmering og matematikk. *Tangenten – tidsskrift for matematikkundervisning, 30*(2), 7–12.

www.caspar.no/tangenten



Figur 4

iterative metoder – da i den forstand at selve problemløsningsprosessen er delt opp i planlagte iterasjoner med delmål og kvalitetssikring av hvert steg. Vi mener da ikke alle iterasjoner som gjøres i en arbeidsøkt, som beskrevet ovenfor, men løsningen av delmål.

Selv vår tilsynelatende enkle oppgave kunne ha to slike delmål:

1 Få hjulet til å rulle med riktig hastighet.
2 Gjøre løsningen uavhengig av hjulets radius.

Hvis vi gjorde dette og innførte en kvalitetssikring av hvert delmål, ville denne sjekken ikke bare se på det visuelle resultatet, men også kunne sjekke en forklaring av hvorfor det var riktig.

Et eksempel på en slik iterativ tilnærming er den metoden som i programmeringsverdenen kalles *Extreme Programming* (https://no.wikipedia.org/wiki/Extreme_Programming). En annen side ved Extreme Programming er beskrivelse av noen roller og samarbeidsformer. Den første er parprogrammering, som sier at all kode skal skrives med to personer foran tastaturet. En skriver, og en kommenterer, stiller spørsmål og kommer med råd. Rollene kan byttes. Den andre er beskrivelsen av kunde/oppdragsgiver. I skolesammenheng er det naturlig å se læreren i rollen som oppdragsgiver, altså den rollen som skal godkjenne det arbeidsgruppa har gjort.

Lærerrollen

Når programmering skal integreres i matematikkfaget, kan det bli en utfordring for læreren, spesielt hvis læreren ikke har noen bakgrunn eller erfaringer med programmering. Slik vi har erfart, vil det være størst behov for lærerens kunnskaper i planleggingsfasen av oppgaven og for å få elevene i gang. Etter det vil lærer-

```
//------------- endre -------------
// Hvor mye skal figuren flytte på seg når den tegnes på nytt
float pe=1.0;     // Posisjons endring
// Hvor mange grader skal figuren rotere når den tegnes på nytt
float re=4;       // Rotasjons endring
//------------- endre ---------
```

Figur 5

```
//---------- endret --------------
float omkrets=2*radius*PI;
// Hvor mye skal figuren flytte på seg når den tegnes på nytt
float pe=1.0;                // Posisjons endring
// Hvor mange grader skal figuren rotere når den tegnes på nytt
float re=pe/(omkrets/360);  // Rotasjons endring
//---------- endret -----------
```

Figur 6

Stenseth, B., Kaufmann, O. T. & Forsström, S. E. (2019). Programmering og matematikk. *Tangenten – tidsskrift for matematikkundervisning, 30*(2), 7–12.

rollen fungere mer som en guide, og læreren vil også ha en viktig rolle når det gjelder elevenes samarbeid. I planleggingsfasen kan det være en utfordring for en lærer å finne riktig vanskegrad i oppgaver som skal fungere både matematisk og programmeringsmessig. Liknende utfordringer blir også trukket fram i andre studier. Bray og Tangney (2017) peker på at programmering som oftest brukes i klasserommet som et redskap for å løse spesifikke matematikkoppgaver, i stedet for å fokusere på arbeidsmåter og metodikk i skjæringspunktet mellom matematikk og programmering. For å overkomme denne utfordringen tenker vi at det er viktig for en lærer å kunne legge opp til en planlagt prosess der eleven sammen med læreren kan kvalitetssikre stegene. Videre må det legges til rette for deling av løsninger og samarbeid mellom elevene.

Referanser

Balanskat, A. & Engelhardt, K. (2015). *Computing our future: Computer programming and coding – Priorities, school curricula and initiatives across Europe*. Brussel European Schoolnet

Bray, A. & Tangney, B. (2017). Technology usage in mathematics education research – A systematic review of recent trends. *Computers & Education, 114*, 255-273.

Grover, S. & Pea, R. (2013). Computational thinking in K—12: A review of the state of the field. *Educational Researcher, 42*(1), 38-43.

Noter

1 Det er viktig å understreke her at disse kompetansemålene er basert på utkastet som er sendt på høring.

2 Vi hadde lagt inn i oppgaven at koden også skulle fungere dersom de endrer radius på figuren.

Stenseth, B., Kaufmann, O. T. & Forsström, S. E. (2019). Programmering og matematikk. *Tangenten – tidsskrift for matematikkundervisning, 30*(2), 7–12.

www.caspar.no/tangenten