Masteroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for samfunns- og utdanningsvitenskap
Institutt for lærerutdanning

Vilde Dahle

Blokkprogrammering som representasjonsregister i matematikk

En kvalitativ designstudie om hvordan elever på 4. trinn bruker blokkprogrammering som semiotisk representasjonsregister for tallfølger

Masteroppgave i matematikkdidaktikk 1.-7. trinn
Veileder: Benedikte Grimeland, Yvonne Grimeland, Oda Tingstad Burheim og Torunn Klemp

Mai 2023

NTNU
Kunnskap for en bedre verden

Vilde Dahle

Blokkprogrammering som representasjonsregister i matematikk

En kvalitativ designstudie om hvordan elever på 4. trinn bruker blokkprogrammering som semiotisk representasjonsregister for tallfølger

Masteroppgave i matematikkdidaktikk 1.-7. trinn
Veileder: Benedikte Grimeland, Yvonne Grimeland, Oda Tingstad Burheim og Torunn Klemp
Mai 2023

Norges teknisk-naturvitenskapelige universitet
Fakultet for samfunns- og utdanningsvitenskap
Institutt for lærerutdanning

NTNU
Kunnskap for en bedre verden

Sammendrag

Denne studien undersøker blokkprogrammering som et semiotisk representasjonsregister med tallfølger som matematisk kontekst. Problemstilling for studien er: *hvordan bruker elever på 4. trinn blokkprogrammering som semiotisk representasjonsregister i arbeid med tallfølger?* Studien er en del av forskningsprosjektet LAB-TEd ved NTNU, som forsker på prosesser knyttet til lærerstudenters forsknings- og utviklingsprosjekter.

For å undersøke blokkprogrammering som semiotisk representasjonsregister, anvendte jeg Duval (2006) sin teori om semiotiske representasjoner. Med støtte i Bråting og Kilhamn (2021), presenterer jeg hvordan blokkprogrammering kan anses som et representasjonsregister for tallfølger. Studien er en kvalitativ designstudie hvor jeg har samlet datamaterialet i to omganger, med justeringer fra første til andre gjennomføring. Datainnsamlingen besto av et undervisningsopplegg utformet etter PRIMM-modellen (Sentance et al., 2019), hvor totalt sju elever deltok. Det kvalitative datamaterialet som er samlet inn består av observasjoner i form av lyd-, video- og skjermopptak fra elevenes samtaler og programmeringsprosess. Analysen tok utgangspunkt i konstant komparativ metode, med en abduktiv tilnærming. Analysen fulgte stegene i konstant komparativ metode, men kategoriene jeg definerte tok utgangspunkt i både datamaterialet og Duvals (2006) teori om semiotiske representasjoner.

Resultater fra studien viser at elevene bruker blokkprogrammering på måter som hindrer dem i å gjøre omdanning til blokkprogrammering som register. I analysen definerte jeg to hovedkategorier: *omdanning mellom ulike registre* og *betingelser i blokkprogrammering som register*. Analysen i studien viser at elevene gjorde omdanning fra blokkprogrammering til naturlig språk, men ikke omvendt. Bruk av blokker som omhandler variabler viste seg å være utfordrende for elevene, i deres forsøk på å representere tallfølger som blokkprogram. I diskusjon av funnene demonstrerer jeg hvordan ulike registre benytter samme tegn, men tildeler tegnene ulik mening. Konflikter ut ifra tegns betydning på tvers av registre, skaper utfordringer ved omdanning.

Abstract

This study investigates block-based programming as a register of semiotic representations with the mathematical context being number sequences. The research question is as follows: *how does fourth grade students use block-based programming as a register of semiotic representations when working with number sequences?* This study is a part of the research project LAB-TEd at NTNU, which studies processes related to student teachers research and development projects.

To examine block-based programming as a register of semiotic representation, I applied Duval's (2006) theory about semiotic representations. With support from Bråting and Kilhamn (2021), I present how block-based programming can be considered a register of representations in the context of number sequences. This study is a qualitative design study where I have collected the data in two iterations, with revision from the first to the second implementation. The data collection consisted of a lesson design, based on the PRIMM model (Sentance et al., 2019), in which a total of seven students participated. The qualitative data that I collected consists of observations in the form of audio, video and screen recordings from the students' conversations and programming process. The analysis was based on the constant comparative method, with an abductive approach. The analysis followed the steps of the constant comparative method, but the categories I defined was both based on the data material and Duval's (2006) theory on semiotic representations.

Results from the study show that the students use block-based programming in ways that prevented them from making a conversion to block-based programming as a register. In the analysis I defined two main categories: *conversion between different registers* and *conditions in block-based programming as a register*. This study's analysis shows that the students made a conversion from block-based programming to natural language, but not the other way around. Using blocks that deal with variables proved to be challenging for the students, in their attempts to represent number sequences as a block program. In discussing the findings, I demonstrate how different registers use the same signs, but assign different meanings to those signs. Conflicts based on the sign's meaning across registers create challenges during conversion.

Forord

For fem år siden begynte jeg på grunnskolelærerutdanning for 1.-7. trinn uvitende om at jeg noen år senere skulle velge å fordype meg i programmering som en del av matematikk-faget. I forkant av dette masteråret kunne jeg svært lite om programmering. Etter utallige timer med opp- og nedturer kan jeg nå si at jeg sitter igjen med både kompetanse i, og engasjement for, programmering i matematikkfaget.

Gjennom LAB-TEd-prosjektet har jeg fått mulighet til å utvikle en masteroppgave som binder sammen forskningsfeltet og undervisningsfeltet. Denne studien gir både lærerstudenter, lærere og lærerutdannere et unikt perspektiv på hvordan blokkprogrammering kan forenes med matematisk aktivitet. Ved å undersøke elevers bruk av blokkprogrammering som et semiotisk representasjonsregister, bidrar denne studien med kunnskap om hvordan blokkprogram kan være en representasjon for matematikk, på linje med en tegning eller en graf.

Jeg ønsker å takke mine veiledere Benedikte, Yvonne, Oda og Torunn for deres iherdige arbeid fra prosjektets start til prosjektets slutt. Dette masterprosjektet har pågått i ett år og veiledningen deres har skapt en avgjørende trygghet i prosessen med masteroppgaven.

Jeg vil også rette en takk til min praksislærer og elevene som har deltatt i denne studien. Det har vært supert å samarbeide med dere!

Sist, men ikke minst, vil jeg rette en ekstra takk til min samboer og private «programmeringskonsulent». Med deg som faglig og emosjonell støtte kan jeg stolt se tilbake på masteroppgaven jeg har utarbeidet.

Trondheim, mai 2023
Vilde Dahle

Innhold

Figurer

Tabeller

1 Innledning

1.1 Deltakelse i et overordnet prosjekt

Denne masteroppgaven er en del av forskningsprosjektet LAB-TEd (Learning, Assessment and Boundary Crossing in Teacher Education). LAB-TEd-prosjektet forsker på samarbeidet mellom universitet, skole og lærerstudenter ved å blant annet studere prosesser knyttet til lærerstudenters forskning- og utviklingsarbeid i skolen (Postholm, 2021). Prosjektet har også som mål å avdekke hindringer i endringsprosessen, som vil komme til nytte ved tilsvarende samarbeids- og utviklingsprosesser i utdanningssystemet.

LAB-TEd er et samarbeid mellom NTNU, UIT og King's College i London. LAB-TEd-prosjektet ved NTNU startet opp høsten 2019 med et kull med fire matematikkstudenter og fire kroppsøvingsstudenter. Min periode i LAB-TEd-prosjektet begynte høsten 2020 og avsluttes sommeren 2023. Jeg er en del av kull 2 av matematikkstudenter. I løpet av denne perioden har jeg og tre andre matematikkstudenter deltatt i alle praksisperioder sammen og ved samme skole hver gang. Vi har hatt tre ulike praksislærere, men de har alle vært tilknyttet prosjektet. Både våre FoU-prosjekter og masteroppgaver er gjenstand for forskning i prosjektet.

Samarbeidsprosessen med å utforme denne masteroppgaven startet omtrent et år før innlevering. Veiledere fra NTNU, praksislærer og oss lærerstudenter gikk i dialog om hva vi som studenter ønsket å forske på, samt hvilken kunnskap skolen hadde behov for. Programmering ble tidlig foreslått som overordnet tema, noe skolen også hadde behov for ytterligere kompetanse i. Praksisperioden på høsten brukte vi som en innledning til datainnsamlingen vi gjorde på våren. Siden vår praksislærer er involvert i LAB-TEd-prosjektet inngikk vi dialog om datainnsamlingen allerede på høsten. Dialogen fortsatte med jevnlige møter utover året helt fram til oppgavens innlevering. Datainnsamlingen foregikk hovedsakelig i praksisperioden og praksislærer var hele veien orientert om hvordan datainnsamlingen foregikk. Gjennom jevnlige møter med veiledere fra NTNU og praksislærer i praksisperioden, har masteroppgaven blitt formet til slik den er nå.

Vi i studentgruppa ble sammen enige om å benytte programmeringsmiljøet MakeCode, i motsetning til det mye brukte programmeringsmiljøet Scratch. Valget falt på MakeCode, fordi MakeCode hører sammen med den fysiske mikrokontrolleren micro:bit (Ball et al., 2019), og fordi alle grunnskoler i Norge har klassesett med micro:bit (Super:bit, u.å.).

1.2 Bakgrunn

I 2020 inntrådte Fagfornyelsen som skolens gjeldende læreplan. For første gang kom programmering på læreplanen fra 4. trinn. For 1. - 7. trinn er programmering kompetansemål i matematikk, naturfag og kunst og håndverk. For å forstå begrunnelsen for innføringen av programmering kan vi se til Ludvigsen-utvalgets nasjonale utredning fra 2015 og utredningen fra 2016 utledet av en ekspertgruppe utnevnt av Utdanningsdirektoratet. Ludvigsen-utvalget etterspurte en fornyelse av skolefagene for å imøtekomme fremtidens behov for kompetanse (NOU 2015: 8). Utvalget anbefalte blant annet fire kompetanseområder for fagfornyelsen: fagspesifikk kompetanse, kompetanse i å lære, kompetanse i å samhandle,

og kompetanse i å utforske og skape (NOU 2015: 8, s. 8). Ifølge Sevik (2016) kan programmering relateres til alle disse kompetanseområdene. Ludvigsen-utvalget nevner ikke programmering, men året etter foreslår et ekspertutvalg utnevnt av Utdanningsdirektoratet å opprette et nytt praktisk skolefag med teknologi og programmering i fokus (Sevik, 2016).

Flere land verden over implementerer programmering i sine læreplaner (Kaufmann & Stenseth, 2021; Kilhamn et al., 2022; Kilhamn & Bråting, 2021). Skoleverkets tilnærming til programmering kan deles inn i tre kategorier: (1) som et eget fag, (2) som et tverrfaglig fokus eller (3) implementert i et allerede etablert fag, eksempelvis matematikk (Bocconi et al., 2018). I mange europeiske land har programmering vært en del av IKT-spesifikke eller rene programmeringsfag (Balanskat & Engelhardt, 2014). I Skandinavia er programmering først og fremst relatert til matematikkfaget (Bocconi et al., 2018). Utdanningsdirektoratet bruker begrepet *algoritmisk tenkning* som argument for innføringen av programmering på barnetrinnet (Utdanningsdirektoratet, 2019). Algoritmisk tenkning kan beskrives som en problemløsningsmetode basert på nedbryting av problemet til mer håndterlige delproblem (Utdanningsdirektoratet, 2019). Algoritmisk tenkning nevnes gjerne i sammenheng med fremtidens kompetansebehov (Bråting & Kilhamn, 2021; Csizmadia et al., 2015; Wing, 2006; Wu & Richards, 2011).

Programmering blir av noen også nevnt som en konkret ferdighet for fremtiden (Bråting & Kilhamn, 2021; Forsström & Kaufmann, 2018; Kaufmann & Stenseth, 2021; Sevik, 2016). I utdanningspolitiske sammenhenger omtales fremtidens ferdigheter ofte som det 21. århundrets ferdigheter (Miller, 2019). Begrepet knyttes blant annet til teknologiske og innovative ferdigheter, slik som programmering. Van Laar et al. (2017) trekker frem blant annet kritisk tenkning, kommunikasjon, samarbeid og kreativitet som karakteristisk for 21. århundrets kompetanse, fordi hun hevder de relaterer mer til dagens sosiale og økonomisk situasjon, enn det forrige århundre.

Forsström og Kaufmann (2018) gjorde en litteraturstudie som undersøkte det vitenskapelige grunnlaget for programmeringens inntreden i matematikkfaget fra perioden 1995 til 2018. De fant at programmering var forbundet med økt motivasjon for noen elevgrupper og økt matematisk prestasjon i noen studier (Forsström & Kaufmann, 2018). Programmering ble også knyttet til samarbeidsbasert læring og læreres nye rolle i undervisning (Forsström & Kaufmann, 2018). Resultatene fra studiene de undersøkte var riktignok ikke generaliserbare. Forsström og Kaufmann (2018) påstår også at forskningen relatert til programmering i matematikkundervisning er varierende og mangelfull.

Duval (2006, 2017) hevder kjernen av all matematisk læring er å gjøre transformasjoner mellom semiotiske representasjoner. En transformasjon mellom semiotiske representasjoner er å kunne representere det samme matematiske objektet på ulike måter (Duval, 2006, 2017). Bråting og Kilhamn (2021, s. 174) har undersøkt programmering i lys av Duvals (2006) teori om semiotiske representasjoner, og hevder programmeringsspråk er et eget semiotisk representasjonsregister. Med andre ord, representasjoner for matematiske objekter kan skapes gjennom programmering. Hvis kjernen av matematisk læring er veksling mellom representasjoner, og programmering kan anses som et register for representasjoner, kan programmering fungere som en ny innfallsvinkel i matematikkundervisning med dagens læreplan.

1.3 Min studie

I denne studien forsker jeg på 4. trinnselever. Jeg tar derfor utgangspunkt i kompetansemålet etter 4. trinn som omhandler programmering: «lage algoritmar og uttrykkje dei ved bruk av variablar, vilkår og lykkjer» (Kunnskapsdepartementet, 2019, s. 8). Undervisningsopplegget i denne studien inkluderer bruk av variabler og løkke, ikke betingelse[1].

Kompetansemålet for 4. trinn stiller ikke krav til matematisk innhold i programmeringen. Kompetansemålet fokuserer på variabel, betingelse og løkke. Dette kan man undervise i og om, uten at programmeringen er tillagt matematisk innhold. Jeg valgte tallfølger som matematisk kontekst for programmeringsaktivitetene, fordi tallfølger ikke er forsket på i sammenheng med blokkprogrammering, samtidig som det er praktisk mulig å lage blokkprogram for tallfølger.

I denne studien tar jeg utgangspunkt i Bråting & Kilhamns (2021, s. 174) påstand om at programmeringspråk er et eget register for semiotiske representasjoner. I teorikapittelet vil jeg presentere Duvals (2006) teori om semiotiske representasjoner og samtidig belyse hvordan blokkprogrammering er et representasjonsregister i arbeid med tallfølger. Ifølge Duval (2006, s. 104) bør forskning på læring av, og utfordringer med, matematikk, undersøkes ved å studere elevers egenproduserte arbeid i faget. Ved å bruke Duvals (2006) perspektiver på semiotiske representasjoner og representasjonsregister undersøker jeg hvordan et utvalg elever på 4. trinn bruker blokkprogrammering som representasjonsregister for tallfølger. Problemstillingen min er derfor:

Hvordan bruker elever på 4. trinn blokkprogrammering som semiotisk representasjonsregister i arbeid med tallfølger?

I et representasjonsregister kan man gjøre både behandling og omdanning (Duval, 2006, 2017). Behandling er veksling mellom representasjoner i samme register og omdanning er veksling mellom representasjoner på tvers av registre (Duval, 2004). I hvilken grad elever gjør behandling og omdanning med blokkprogrammering som register, faller innenfor det jeg kaller å *bruke* blokkprogrammering som register. Elevenes konkrete tolkning og bruk av blokkene inkluderer jeg også som bruk av blokkprogrammering, fordi Duval (2006, 2017) påpeker at alle register inneholder tegn som bare fungerer i eget register, og som først får en funksjon i registeret når det samhandler med andre tegn. En enkeltblokk i blokkprogrammering kan anses som et tegn i blokkprogrammering som register, fordi en programmeringsblokk ikke hører hjemme i andre registre og dessuten kun gir mening i blokkprogrammering når den bygges sammen med andre blokker.

[1] Vilkår på nynorsk

2 Teori

Denne studien undersøker bruk av blokkprogrammering basert på et konstruktivistisk syn på læring, hvor elevaktiv undervisning og dialog er sentralt (Bada, 2015). For å kunne ta stilling til bruk av blokkprogrammering som register, er det nødvendig å forstå hva programmering og blokkprogrammering er. Jeg benytter Bueie (2019) sin definisjon av programmering og utdyper konseptet basert på betraktninger fra Weintrop og Wilensky (2017), Franklin et al. (2016), Bell et al. (2009) og Wohl et al. (2015). MakeCode introduseres som blokkbasert programmeringsmiljø ved Austin et al. (2020).

Tallfølger som mønster er den matematiske konteksten for programmeringsaktivitetene i denne studien. Begrepene mønster og struktur presenteres ut ifra Wijns et al. (2019) og Mulligan og Mitchelmore (2013) sine definisjoner. Tallfølger som mønster utdypes ved Adams og Essex (2013). Min studie anvender Duvals (2006) teori om semiotiske representasjonsregistre for å studere elevers bruk av blokkprogrammering. Duvals (2006) teori om semiotiske representasjoner blir grundig beskrevet, inkludert hvordan blokkprogrammering kan anses som et eget register i arbeid med tallfølger. Betraktninger fra Bråting og Kilhamn (2021) og Kilhamn et al. (2022) benyttes for å supplere Duvals (2006) teori.

Avslutningsvis vil jeg presentere et utvalg tidligere forskning på programmeringsundervisning i matematikkfaget. Utvalget baserer seg på en litteraturstudie av Forsström & Kaufmann (2018), samt tilleggslitteratur som er publisert etter 2018.

2.1 Konstruktivistisk syn på læring

Denne studien baserer seg på et konstruktivistisk syn på læring. Konstruktivisme har røtter i flere læringsteorier som anser læring som tilegnelse av erfaringer (Bada, 2015). Konstruktivistisk læringssyn kan komme til uttrykk på ulike måter, men noen sentrale ideer er gjennomgående i dette perspektivet på læring (Bada, 2015). Læring anses som en prosess som skjer kognitivt hos individet, hvor individet vurderer nye erfaringer opp mot tidligere erfaringer (Bada, 2015). Individet anses å ha en aktiv rolle i egen kunnskapsutvikling. Som begrepet konstruktivisme antyder er kunnskap noe som *konstrueres*. Helt spesifikt konstrueres kunnskap kognitivt hos individet, men gjerne i samspill med andre (Bada, 2015). I samspill med andre kan man diskutere, reflektere og problematisere både tidligere erfaringer og nye erfaringer. Et slikt sosialt samspill fasiliterer for individuell refleksjon som leder mot konstruksjon av ny kunnskap (Bada, 2015).

Undervisning med et konstruktivistisk syn på læring vil blant annet vektlegge elevdeltakelse, elevsamarbeid og en deltakende lærerrolle som fasiliterer dialog (Bada, 2015). Oppsummert vil undervisning basert på konstruktivisme være dynamisk og opptatt av prosessen underveis i læringsarbeidet. Prosessen som foregår anses som selve læringen, fordi det er da ny kunnskap konstrueres (Bada, 2015).

2.2 Programmering i matematikkfaget

I læreplanen for Kunnskapsløftet 2020 har programmering blitt en del av matematikkfaget. Fra og med 4. trinn inngår programmering som ett kompetansemål ved alle trinn i grunn-

skolen (Kunnskapsdepartementet, 2019). Argumentet for å introdusere programmering på barnetrinnet knyttes til begrepet algoritmisk tenkning (Utdanningsdirektoratet, 2019). Det finnes varierte beskrivelser av algoritmisk tenkning, men Utdanningsdirektoratet beskriver det som en problemløsningsmetode hvor et problem ansees som en rekke delproblem som kan løses stegvis (Utdanningsdirektoratet, 2019). De trekker også fram at det krever en systematisk og analytisk tilnærming for å løse hvert enkelt delproblem, samt en utholdenhet til å kunne prøve og feile. Ifølge Wing (2006) er algoritmisk tenkning en ferdighet for alle og enhver, ikke en ferdighet utelukkende for informatikere. Algoritmisk tenkning blir av Wu & Richards (2011) beskrevet som en digital «literacy», en ferdighet alle behøver for å mestre det stadig mer digitaliserte samfunnet. Digitale ferdigheter er også en av dem fem grunnleggende ferdighetene i LK20, på tvers av fagene (Kunnskapsdepartementet, 2017).

Programmering kan forstås som en aktivitet hvor man skriver detaljerte instruksjoner på et språk som en datamaskin kan lese (Bueie, 2019, s. 22). Bueie (2019) benytter ordet *program* når han refererer til en samling instruksjoner. Kaufmann et al. (2018, s. 75) skriver at programmering typisk omtales som *koding* i skolesammenheng. På samme måte kan også et program omtales som en *kode*. Jeg vil forholde meg til Bueies definisjon (2019, s. 22) og bruke begrepene program og programmering slik han definerer dem.

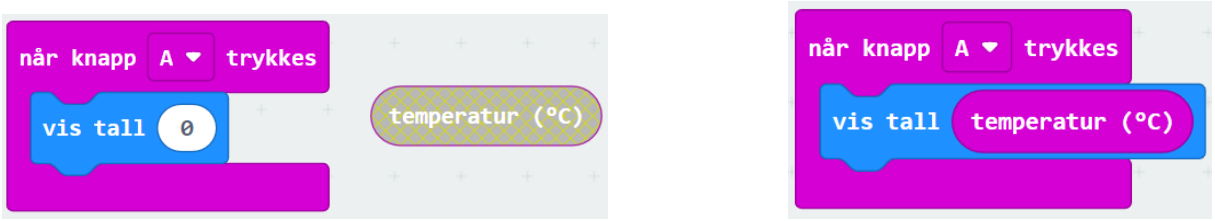
Overordnet kan programmering deles inn i tekstbasert og blokkbasert språk. Tekstbasert programmeringsspråk er den tradisjonelle måten å gi en datamaskin instruksjoner på. Tekstbaserte programmeringsspråk krever at den som programmerer selv må kjenne til språkets kommandoer og hvordan disse kan kombineres for å lage et fungerende program (Franklin et al., 2016). Blokkbaserte programmeringsspråk består av ferdiglagde blokker som skal settes sammen for å oppnå et fungerende program (Weintrop & Wilensky, 2017). Blokkprogrammering ble utviklet for å gi elever muligheten til å lære grunnleggende konsepter ved programmering, uten byrden av å memorere kommandoer (Franklin et al. 2016, s. 217).

Programmering med blokkbasert programmeringsspråk kan foregå på ulike plattformer eller *programmeringsmiljø*. Weintrop & Wilensky (2017) benytter ordet programmeringsmiljø når det er snakk om hele plattformen man programmerer i, ikke bare blokkene og hvordan man bruker dem. Jeg vil også omtale en programmeringsplattform som et programmeringsmiljø, for å unngå at begrepet program får ulike betydninger.

I blokkbaserte programmeringsmiljø, slik som Scratch eller MakeCode, er blokkene bevisst utformet med spesifikke former og farger (se figur 2.1). Figur 2.1a og 2.1b viser tre ulike blokker med ulik farge og form. Blokkenes form indikerer hva som er gyldig syntaks (Weintrop & Wilensky, 2017). Syntaks handler om regler i språket, hvilke ord man kan bruke og på hvilken måte disse ordene kan kombineres for å skape større enheter, typisk setninger (Istad & Kristoffersen, 2019). Med andre ord vil blokkenes former veilede elevene ved å indikere hvilke blokker som kan kombineres og ikke. Blokken som representerer temperatur i figur 2.1 er avrundet og vil bare kunne settes inn i en annen blokk som har avrundet form. Om man forsøker å plassere to blokker sammen, som syntaktisk ikke er gyldig, vil blokkenes former forhindre at de settes sammen (Weintrop & Wilensky, 2017).

Som vi ser i figur 2.1a blir temperaturblokken «sladdet» når den står alene. Dette er et eksempel på hvordan blokkbaserte programmeringsmiljø gir programmereren visuell veiledning med utgangspunkt i syntaks. Sladdingen forteller oss at blokken ikke vil fungere

slik den står nå. En blokk som plasseres alene vil sladdes, og regnes ikke som en del av programmet når programmet kjøres.



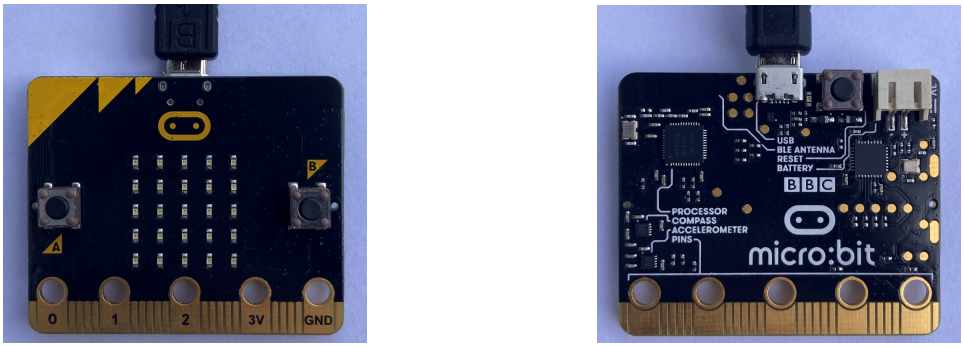
(a) (b)

Figur 2.1: Eksempel på blokker som danner et program

I tillegg til tekstbasert og blokkbasert programmering finnes det et tredje alternativ: analog programmering. Analog programmering er aktiviteter som ikke inkluderer en datamaskin, hvor elevene selv fungerer som algoritmer (Bell, et al., 2009; Wohl et al., 2015). Et eksempel er å følge en oppskrift når man baker kake. Baking krever ingen datamaskin og det er elevene som utfører handlingen. Basert på oppskriften må elevene stegvis ta seg gjennom instruksjonene i kronologisk rekkefølge. For å oppnå en smakfull kake må oppskriften følges nøyaktig og i riktig rekkefølge. Ved å hoppe over steg eller gjøre stegene i feil rekkefølge vil ikke resultatet bli det samme. Hensikten med analoge programmeringsaktiviteter er å lære elevene problemløsning for å oppnå et bestemt mål, samtidig som de får erfaring med grunnleggende ideer ved programmering (Bell et al., 2009). I kake-eksemplet er målet å lage en smakfull kake, samtidig som elevene erfarer betydningen av nøyaktige og kronologiske instruksjoner.

2.2.1 MakeCode

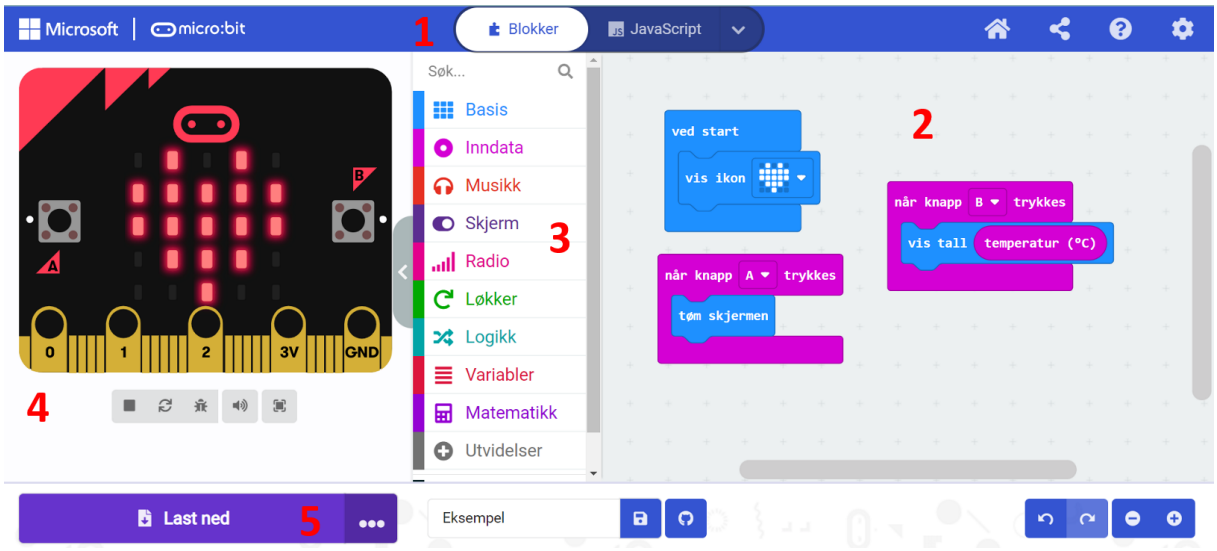
MakeCode er et programmeringsmiljø utviklet av Microsoft. MakeCode tillater programmering med tre programmeringsspråk: blokkbasert, Javascript og Python. JavaScript og Python er tekstbaserte programmeringsspråk. Det er mulig å skrive et program i ett av de tre språkene og deretter oversette det til et av de andre språkene (Austin et al., 2020). Programmering i MakeCode gjør det mulig å programmere BBC sin fysiske mikrokontroller, micro:bit (Ball et al., 2019). Micro:bit er en fysisk mikrokontroller utviklet av BBC for bruk i programmeringsundervisning i Storbritannia (Seneviratne, 2018), se figur 2.2.



(a) Framside (b) Bakside

Figur 2.2: micro:bit 1. generasjon

Programmer laget i MakeCode kan via USB-tilkobling utføres fysisk på en micro:bit. Figur 2.2a viser micro:bit'n sin skjerm og knapper (A og B). 25 lysdioder i midten av micro:bit'n utgjør skjermen. På baksiden av en micro:bit kan man blant annet tilbakestille programmet som kjøres ved å trykke på den svarte knappen øverst ved USB-tilkoblingen. Det er mulig å teste programmet man har lagd i MakeCode, uten å bruke en micro:bit. MakeCode har en innebygd simulator som er en replika av micro:bit'n (Austin et al., 2020). Figur 2.3 viser MakeCode i sin helhet.



Figur 2.3: Programmeringsmiljøet MakeCode

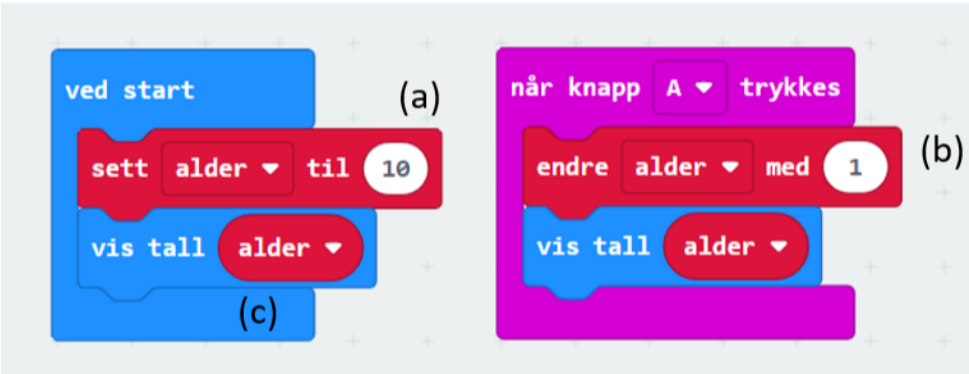
Øverst (1) kan vi se at det er mulig å velge mellom blokker og JavaScript, Python er også et alternativ. Har man lagd et blokkprogram kan man trykke på JavaScript å få det samme programmet vist i JavaScripts tekstbaserte språk. Selve programmet bygges i brukergrensesnittet (2) ved å dra inn blokker fra margen (3). I margen er blokkene blant annet kategorisert etter egenskaper ved micro:bit'n (Austin et al., 2020), slik som musikkavspilling, skjerm og inndata (knapper). Når man har bygd ferdig et program kan man teste programmet gjennom en simulator (4) som er en replika av micro:bit'n. Simulatoren kan spille av lyd, motta knappetrykk og alt annet som den fysiske micro:bit'n kan gjøre. Om man ønsker å kjøre programmet på den fysiske micro:bit'n må man laste ned programmet som en hex-fil (5). Hex-fila gjør det mulig for micro:bit'n å lese programmet man lagde. For at micro:bit'n skal motta informasjonen i programmet må micro:bit'n kobles med USB til datamaskinen. Deretter må hex-fila overføres til micro:bit'n via USB-tilkoblingen (Austin et al., 2020).

Fra margen (3) i MakeCode kan vi blant annet se «løkker». En løkke er en måte å uttrykke gjentakelse på i programmeringsspråk (Istad & Kristoffersen, 2019). I figur 2.4 ser vi bruk av en løkke, hvor innholdet i løkka skal gjenta seg fem ganger. Den grønne løkkeblokka fylles med innhold, og når programmet kjøres vil datamaskinen lese programmet fra topp til bunn. Men i stedet for å stoppe når den kommer til bunn, vil den gjenta sekvensen å lese fra topp til bunn igjen, totalt fem ganger. Hvor mange ganger man ønsker at innholdet skal gjenta seg kan man selv definere. Figur 2.4 demonstrerer et blinkende hjerte på simulatoren eller micro:bit'n sin skjerm.

Figur 2.4: Eksempel på bruk av løkke i MakeCode

Variabel er også noe man kan velge fra margen (3) i MakeCode. Variabler kan man bruke i både algebra og programmering, men begrepet har ulik betydning i de ulike sammenhengene (Kilhamn et al., 2022). Programmeringsvariabel forstås som en navngitt verdi, hvor variabelens navn fungerer som et minne og variabelens verdi er innholdet i minne (Usiskin, 1988). Den definerte variabelverdien lagres i minne, men verdien kan alltid endres utover i programmet (Kilhamn et al., 2022).

I figur 2.5 ser vi eksempel på variabelbruk i et program skapt i MakeCode. Programmet gjør at tallet 10 vises med en gang programmet kjøres, og for hver gang knapp A trykkes så øker alderen med 1, og den nye oppdaterte verdien vises på skjermen. Figuren inkluderer alle de tre variabelblokkene som fins i MakeCode (a, b, c). I eksemplet ser vi en variabel med navn «alder» som er definerte til å være tallverdien 10, ved hjelp av en *settblokk* (a). Ved (b) ser vi en *endreblokk* som endrer alderen med 1. Å «endre» i MakeCode er alltid addisjon, altså endres alderen ved å øke med 1. Trykker man på knapp A én gang, blir den nye aldersverdien 11.



Figur 2.5: Tre varianter av variabelblokker i MakeCode

«Vis tall» er en blokk som vil vise et gitt tall på skjermen. I eksemplet er den tredje variabelvarianten (c) plassert inn i vis tall-blokka. I tillegg til settblokk og endreblokk er det alltid mulig å bruke variabelen alene, slik vi ser i (c). Denne avrundede blokka vil jeg framover omtale som *variabelblokk*, fordi de to andre (a, b) omtales som henholdsvis settblokk og endreblokk. Variabelblokka vil oppdatere sin tallverdi i takt med antall knappetrykk, noe

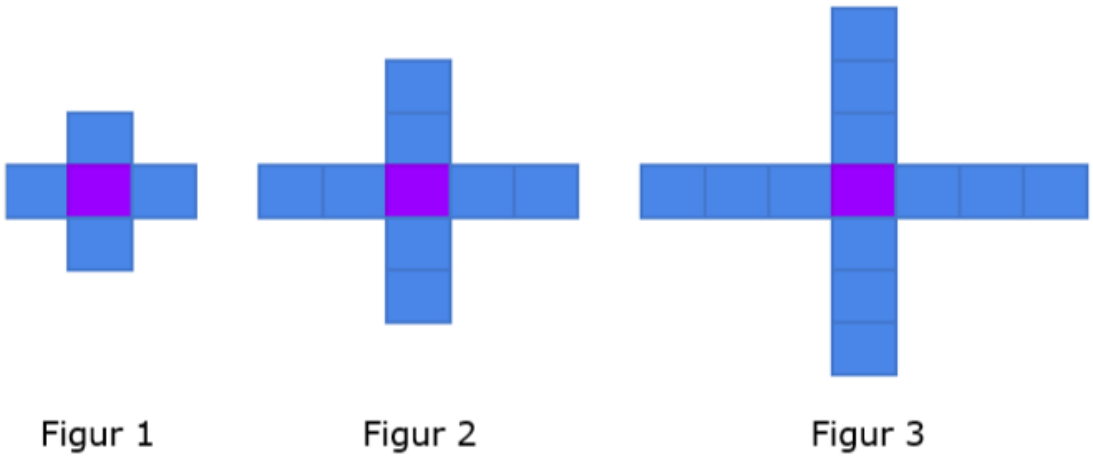
som betyr at selv om «alder» er satt inn i både «ved start» og «knapp A», så vil ulike tallverdier vises, fordi variabelen «alder» oppdaterer sin tallverdi underveis ved trykk på knapp A.

2.3 Mønster i matematikk

Matematiske mønster kan forklares som sammensetninger av enheter, som tall eller objekter, hvor enheten er ordnet regelmessig (Mulligan & Mitchelmore, 2013; Wijns et al., 2019). O, X, O, X, O, X, er et eksempel på en regelmessig ordning av symbolene O og X. De gjentar seg annen hver gang. I litteraturen skilles det mellom repeterende og voksende mønster.

Repeterende mønster beskriver sekvenser av elementer som gjentar seg regelmessig (Papic & Mulligan, 2007). I eksemplet ovenfor er O og X elementer som gjentar seg som sekvenser av OX. Elevers gjenkjenning av en repeterende sekvens, for eksempel OX, er avgjørende for å kunne generalisere og forutse mønster (Wijns et al., 2019). Wijns et al. (2019) nevner at mønsteraktiviteter kan utformes på ulike måter. For eksempel kan oppgaven være å fortsette mønsteret, beskrive en manglende del av et mønster eller å kopiere mønsteret (Wijns et al., 2019).

Voksende mønster beskrives som sekvenser med regelmessig økning eller redusering (Papic & Mulligan, 2007, s. 594). Figurmønsteret i figur 2.6 er et eksempel på et voksende mønster med regelmessig økning. Figurmønsteret har en regelmessig økning på fire. For hver figur øker neste figur med fire klosser, en på hver «arm».



Figur 2.6: Figurmønster som et voksende mønster

Et eksempel på et voksende mønster som ikke øker, men reduserer systematisk, er tallfølgen $\{2, 1, \frac{1}{2}, \frac{1}{4}, ...\}$. Tallfølgen har en systematisk redusering som kan forklares med at hvert neste tall er en halvering av det forrige. For å oppsummere repeterende og voksende mønster, kan man si at et repeterende mønster har enheter som repeterer regelmessig, mens et voksende mønster har en økning eller redusering som repeterer regelmessig.

Mulligan & Mitchelmore (2013) skiller mellom mønster og struktur. De beskriver struktur som på hvilke måter enhetene i mønsteret er organisert og hvordan enhetene relaterer til

hverandre (Mulligan & Mitchelmore, 2013, s. 30). Figurmønsteret i figur 2.6 er i dette eksemplet organisert med en kloss i midten og fire armer i hver himmelretning. Den bestemte strukturen eller organiseringen fungerer som en ramme for hvordan vi kan forvente at neste figur vil se ut. Figurmønsteret kan også beskrives med funksjonsuttrykket $a_n = 4 \cdot n + 1$. Funksjonsuttrykket beskriver strukturen i mønsteret, fordi den uttrykker hvordan enhetene relaterer til hverandre. Ifølge Mulligan og Mitchelmore (2013) er bevissthet omkring mønster og mønsterstruktur avgjørende for de yngste elevenes utvikling i matematikk.

2.3.1 Tallfølger som mønster

En tallfølge er «en ordnet liste som består av et første element, men ikke et siste» (Adams & Essex, 2013, s. 496, egen oversettelse). En ordnet liste av tall hvor fortsettelsen av listen er åpenbar, etterfølges med «...» (Adams & Essex, 2013). En liste med partall: 2, 4, 6, 8, har en tilsynelatende åpenbar fortsettelse. Partallsfølgen uttrykkes derfor slik: {2, 4, 6, 8, ...}.

Tallfølger inkludere ikke en figur, slik som et figurmønster. Ifølge Stylianides (2008) vil det ikke være matematisk mulig å være helt sikker på en tallfølges fortsettelse, fordi den mangler tilhørende figur. Tallfølgen {1, 2, 3, ...} kan for eksempel fortsette som 1, 2, 3 på nytt eller med 4, 5, 6. Tallfølgen {1, 2, 3, ...} kan være både repeterende og voksende, men hvis en definerer den som voksende vil fortsettelsen være 4, 5, 6. Hvis tallfølgens mønster ikke spesifiseres som enten voksende eller repeterende vil elever potensielt fortsette tallfølger ulikt. I slike tilfeller kan det tenkes at barn vil velge det de selv mener er den meste naturlige eller logiske fortsettelsen, basert på tallene man har (Zazkis & Liljedahl, 2002 i Stylianides, 2008).

Strukturen til tallfølger kan uttrykkes som funksjoner (Adams & Essex, 2013; Mulligan & Mitchelmore, 2013). En funksjon er en algebraisk beskrivelse av hvordan to eller flere verdier relaterer til hverandre (Blanton, 2008, s. 31). Tallfølgen bestående av kvadrattallene: {1, 4, 9, 16, 25, ...} kan beskrives med funksjonsuttrykket $a_n = n^2$, hvor n er nummeret i følgen og a er det tilhørende tallet i følgen.

Tallfølger som uttrykkes som funksjoner kan beskrives enten rekursivt eller eksplisitt (Adams & Essex, 2013). Funksjonen $a_n = n^2$ er en eksplisitt beskrivelse av tallfølgen, fordi den uttrykker forholdet mellom a og n uansett verdier av n (Adams & Essex, 2013).

Tallfølgen {1, 3, 7, 13, 21, ...} kan uttrykkes med en rekursiv funksjon, en funksjon som baserer seg på verdien av foregående tall for å beregne neste (Adams & Essex, 2013). Tallfølgen uttrykkes rekursivt, slik: $a_n = a_{n-1} + 2n$, der n er nummeret i følgen og $n - 1$ refererer til det forrige nummeret i følgen. a_{n-1} er derfor det forrige tallet i følgen, før a_n.

2.4 Representasjoner i matematikk

Denne studien undersøker hvordan blokkprogrammering kan brukes som et semiotisk representasjonsregister i arbeid med tallfølger. Begrepet semiotisk representasjonsregister stammer fra Duvals (2006) rammeverk om semiotiske representasjoner. Jeg tar utgangspunkt i to av Duvals (2006) teorier: hans inndeling i fire representasjonsregister, samt behandling og omdanning mellom semiotiske representasjoner. Jeg vil nå presentere disse

teoriene.

Duval (2006, s. 107) hevder at matematikk skiller seg fra andre fagdisipliner, fordi den eneste måten å få tilgang til matematiske objekter på, er via tegn eller semiotiske representasjoner. I for eksempel naturvitenskapelige fagdisipliner er objekter observerbare, enten ved det blotte øyet eller ved hjelp av instrumenter. I matematikk må man ta i bruk konkreter eller representasjoner for å virkeliggjøre abstrakte ideer (Duval, 2006). En representasjon i matematikk «skjuler» en matematisk betydning, noe som gjør representasjonen semiotisk (Duval, 2006, 2017). Semiotikk omhandler tegn som bærer en mening, hvor meningen skiller seg fra tegnet i seg selv (Duval, 2006, 2017). Bruken eller tolkningen av tegnet er også en del av semiotikk. For å eksemplifisere en semiotisk representasjon vil jeg illustrere det matematiske konseptet halvering. Halvering blir forståelig om vi skisserer en sirkel og en strek som deler sirkelen i to like deler. Sirkelen og streken er tegn som sammen utgjør en representasjon av konseptet halvering. Sirkelen som deles i to er ikke bare en sirkel og en strek, det er en meningsbærende matematisk representasjon for halvering. Når Duval (2004, 2006, 2017) omtaler representasjoner er det alltid som semiotiske representasjoner. Semiotiske representasjoner har interne betingelser eller regler som skiller seg fra andre semiotiske representasjoner. Enhver semiotisk representasjon har også ulike måter å skille mellom de meningsbærende enhetene (Duval, 2017, s. 18).

Det matematiske konseptet dobling kan representeres som både $2x$ og $x + x$. Til tross for at representasjonene representerer det samme konseptet, er ikke representasjonene like. Representasjonene er ulike, fordi de benytter ulike symboler og ulik sammensetning av symbolene. Hvilke måter symbolene kan settes sammen på, avhenger av de interne betingelsene som gjelder i hvert register (Duval, 2017, s. 18). I algebra er for eksempel en betingelse at tall eller bokstaver som står inntil et annet tall eller bokstav ($2x$), betyr multiplikasjon. $2x$ refererer til dobling som en multiplikativ prosess, mens $x + x$ refererer til dobling som gjentatt addisjon. Eksempelet viser hvordan to algebraiske representasjoner kan demonstrere ulike egenskaper ved konseptet som de representerer, ved å kombinere ulike symboler, basert på reglene som er betinget i algebra.

I forbindelse med semiotiske representasjoner benytter Duval (2004, 2006, 2017) også begrepet semiotiske representasjonssystemer. Ifølge Duval (2006) vil alle semiotiske representasjoner tilhøre et semiotisk representasjonssystem. For å forstå kompleksiteten ved matematiske tankeprosesser, er det ifølge Duval (2006) nødvendig å sammenligne systemene som blir tatt i bruk. Bråting & Kilhamn (2021, s. 174) benytter Duvals rammeverk og presiserer at de ulike representasjonssystemene demonstrerer ulike egenskaper ved det matematiske objektet som representeres. De eksemplifiserer ved å sammenligne graf og dens tilhørende likning. En likning uttrykker eksplisitt verdienes stigning, mens grafen gir en visuell framstilling av verdienes utvikling.

Begrepene representasjonssystem og representasjonsregister benyttes ofte om hverandre, men de skal ikke forstås helt likt. Duval (2006, s. 111) poengterer at et system kan være et register hvis systemet tillater veksling mellom ulike representasjoner. Duval (2006) har utviklet et rammeverk som han mener inkluderer de fire systemene, men også registrene, som benyttes i matematiske prosesser. I figur 2.7 har jeg gjengitt Duvals (2006) rammeverk om representasjonsregister. Videre vil jeg kun benytte ordet representasjon, med den forutsetning at jeg alltid mener semiotisk representasjon. Det samme vil gjelde de øvrige

begrepene representasjonssystem og representasjonsregister, de vil alltid være semiotiske.

	Diskursive representasjoner	Ikke-diskursive representasjoner
Multi-funksjonelle register	**Naturlig språk** Muntlig og skriftlig	**Ikonisk system:** figurer, skisser, mønster **Ikke-ikonisk system:** geometriske figurer som kan bli konstruert med hjelpemidler
Mono-funksjonelle register	**Symbolsk system** Algebraisk notasjon, likninger, funksjoner etc.	**Kartesiske grafer og diagrammer** Flerdimensjonale figurer: tallinje, koordinatsystem

Figur 2.7: Duvals inndeling i semiotiske representasjonsregister

(Duval, 2006, s. 110, egen oversettelse)

Figur 2.7 viser en oversikt over alle representasjonsregistrene. *Naturlig språk* er registeret vi er i når vi omtaler matematiske objekter med ord, enten muntlig eller skriftlig (Duval, 2006). *Det symbolske systemet*, eller registeret, omhandler representasjoner som benytter symboler og matematiske tegn for å representere et matematisk objekt (Duval, 2006). Eksempelet med dobling representert som $2x$ og $x + x$, er representasjoner som tilhører det symbolske registeret, fordi det benyttes symboler og matematiske tegn for å uttrykke konseptet dobling. Dobling kan også representeres som en graf i registeret som omhandler *flerdimensjonale figurer*. Dobling kan for eksempel også representeres *ikonisk* som et figurmønster. Se figur 2.6 for eksempel på et figurmønster.

I matematikk tenker vi aldri utelukkende i ett register, vi befinner oss i flere på samme tid (Duval, 2017, s. 83). Bråting & Kilhamn (2021) konkretiserer Duvals betraktning og skriver at elever i arbeid med matematikk ofte tar i bruk minst to registre, hvor det ene gjerne er naturlig språk.

Duvals (2006, 2017) representasjonsregistre kan deles inn etter hvilke funksjoner de fyller. Monofunksjonelle registre, som symbolsk register og flerdimensjonale figurer, har kun en funksjon i matematikkfaget (Duval, 2017, s. 84). Multifunksjonelle registre, som språk, mønster og figurer, kan benyttes på flere områder, for eksempel i kommunikasjon eller i kreative prosesser (Duval, 2017, s. 84).

Register kategoriseres også som enten diskursive eller ikke-diskursive. Diskursive registre kategoriseres ved at de produserer og organiserer sekvenser av ord og symboler (Duval, 2017, s. 85). Ikke-diskursive registre omhandler flerdimensjonale figurer og former, ikke bruk av ord og symboler for å uttrykke mening (Duval, 2006, 2017).

Representasjonsregistre er avgjørende i matematiske aktiviteter, fordi all matematisk prosess krever en veksling mellom ulike semiotiske representasjoner (Duval, 2006), enten ved at vekslingen skjer innad i et register eller på tvers av registre. En veksling innad i samme register kalles «treatment» (Duval, 2004), eller behandling på norsk.

Et eksempel på behandling av en representasjon for tallfølger er:

$$\{2, 4, 6, 8, 10, ...\} \tag{1}$$

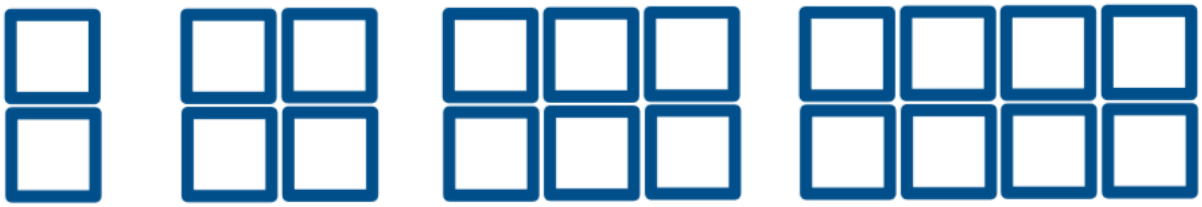
$$\{2 \cdot 1, 2 \cdot 2, 2 \cdot 3, 2 \cdot 4, 2 \cdot 5, ...\} \tag{2}$$

$$a_n = 2n, \tag{3}$$

der a er tall i følgen og n er tallnummeret i følgen.

Eksemplet viser en partallsfølge (1) som blir omgjort til en representasjon som konkretiserer at økningen baserer seg på at tallet 2 multipliseres med tallnummeret i følgen (2), og deretter omgjøres til den eksplisitte beskrivelsen $2n$ (3). Disse vekslingene mellom representasjoner er et eksempel på behandling, fordi alle representasjonene er en del av det samme registeret, symbolsk register.

Når veksling mellom representasjoner foregår på tvers av representasjonsregistre kalles det «conversion» (Duval, 2004), eller omdanning. Tar vi utgangspunkt i eksemplet på behandling kan et eksempel på omdanning være fra {2, 4, 6, 8, 10, ...} til naturlig språk «alle tallene er partall». Partallsfølgen kan også illustreres som et visuelt mønster, slik vi ser i figur 2.8. Figuren viser partallsfølgen i det ikoniske registeret.



Figur 2.8: Ikonisk representasjon av partallsfølgen

Omdanning er en kognitivt krevende oppgave for elever på alle trinn i skolesystemet (Duval, 2006). Omdanning er mer krevende enn behandling, fordi ethvert registerskifte krever at man gjenkjenner det matematiske objektet i begge representasjonene, til tross for at de ulike representasjonene ofte har lite til felles. Hvor krevende omdanningen er avhenger av hvilke registre omdanningen består av (Duval, 2006, s. 112).

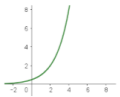
Duval (2006) kategoriserer omdanning i to typer: *kongruent* og *ikke-kongruent*. Kongruent omdanning forekommer når representasjonen man gjør omdanning til sammenfaller fullstendig med representasjonen man gjorde omdanningen fra (Duval, 2006). Bråting & Kilhamn (2021, s. 174) gir følgende eksempel: «fem pluss fire er lik ni» som naturlig språk, sammenfaller fullstendig med den symbolske representasjonen 5+4=9. Tilsvarende vil en ikke-kongruent omdanning være å si «ni er summen av 4 og 5». I dette tilfelle benyttes ikke ordet pluss, og rekkefølgen på ordene sammenfaller ikke med den symbolske representasjonen 5+4=9. Ikke-kongruent omdanning er utfordrende for elever som lærer matematikk (Duval, 2006).

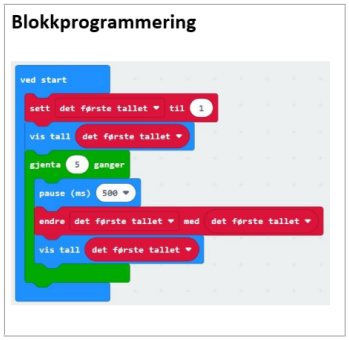
2.4.1 Blokkprogrammering som representasjonsregister for tallfølger

For å kunne kalle blokkprogrammering et eget register, må programmeringsmiljøet muliggjøre semiotiske representasjoner av matematiske objekter, for eksempel tallfølger. Blokk-

programmering inneholder blokker som kan settes sammen og danne et program. Blokkprogram kan opptre som semiotiske representasjoner hvis de representerer et matematisk konsept eller objekt. Figur 2.11 er et eksempel på et fungerende program som gir tallfølgen {1, 2, 4, 8, 16, ...} på skjermen. Øker man verdien som bestemmer antall ganger løkka skal gjenta seg, vil tallfølgen fortsette lengre. Det er mulig å lage program som representerer andre matematiske objekter, og program som ikke representerer matematikk i det hele tatt. Av den grunn vil ikke blokkprogrammering være et semiotisk representasjonsregister i alle sammenhenger.

For å tydeliggjøre hvordan blokkprogrammering skiller seg fra de andre registrene har jeg utformet en figur (figur 2.9) som eksemplifiserer tallfølgen {1, 2, 4, 8, 16, ...} i alle registrene, inkludert blokkprogrammering. Figuren baserer seg på Duvals (2006) rammeverk for semiotiske representasjonsregistre.

	Diskursive representasjoner	Ikke-diskursive representasjoner
Multi-funksjonelle register	**Naturlig språk (muntlig eller skriftlig)** "Tallene dobler seg for hver gang"	**Ikonisk system:** som et figurmønster **Ikke-ikonisk system:** geometriske figurer som kan bli konstruert med hjelpemidler
Mono-funksjonelle register	**Symbolsk system** 1, 2, 4, 8, 16... eller $a_n = 2^{n-1}$ $a_n = a_{n-1} + a_{n-1}$ $a_n = 2 \cdot a_{n-1}$	**Kartesiske grafer og diagrammer** Som graf: 



Figur 2.9: Tallfølgen {1, 2, 4, 8, 16, ...} i alle registre

(Duval, 2006, s. 110, egen oversettelse og egne forslag)

Bråting & Kilhamn (2021) hevder at programmeringsspråk burde være et eget representasjonssystem, basert på sammenligninger de gjorde mellom algebraisk notasjon og programmeringsspråk. De poengterer blant annet at begrepene variabel, likhet og algoritme har forskjellig betydning innenfor algebra og programmering, selv om begge benytter de samme ordene. Begrepene variabel, likhet og algoritme anses å være tegn i semiotiske representasjoner, fordi de må settes sammen med andre meningsbærende enheter for å skape en matematisk representasjon (Duval, 2017). Et viktig aspekt ved Duvals (2006) rammeverk om representasjonsregistre er at tegn bare kan fungere i sitt eget register. Siden begrepene variabel, likhet og algoritme bærer ulik mening i de ulike registrene, kan ikke blokkprogrammering være en del av det symbolske registeret. Kilhamn et al. (2022) tydeliggjør hvordan likhetstegnet har ulike betydninger i algebra og programmering med likningen $x = x + 2$. Påstanden er ugyldig som algebraisk notasjon, men gyldig i programmeringssammenheng.

Blokkprogrammering kan ikke anses å være en del av de ikke-diskursive registrene. Blokkprogrammering benytter tekst og tallverdier i sine representasjoner, noe som ikke forekommer i de ikke-diskursive registrene. Ikoniske representasjoner har en visuell framstilling av objektet. Et blokkprogram er ikke en visuell representasjon av tallfølger, fordi det består av blokker, tekst og tallverdier som sammen utgjør en representasjon av tallfølger. Fra figur 2.9 ser vi tallfølgen illustrert som et figurmønster i det ikoniske registeret. Blokkprogram-

mering kan heller ikke regnes som et ikke-ikonisk register, fordi det ikke er en geometrisk figur. Grafer og diagrammer er heller ikke det samme som blokkprogrammering.

Blokkprogrammering kan relateres til de diskursive registrene, fordi det benytter naturlig språk i blokkene og fordi blokkprogram som uttrykker tallfølger har likhetstrekk med tallfølger som funksjonsuttrykk, se figur 2.9. Men ideen om at blokkprogrammering deler egenskaper med to registre er problematisk. Duval (2006, 2017) skriver at hvert register har sine tegn, og interne betingelser knyttet til disse, som kun gjelder i eget register. Gazoni (2018, henvist i Bråting & Kilhamn, 2021) beskriver forskjeller mellom naturlig språk og programmeringsspråk og kommenterer at en vesentlig semiotisk forskjell er hvordan programmeringsspråk ikke tillater nyanser og tvetydighet. Blokkbaserte programmeringsmiljø baserer seg på en syntaks som blokkenes former gir indikasjoner på (Weintrop & Wilensky, 2017). Blokkenes form veileder elevene mot hvilke sammensetninger av blokker som programmeringsmiljøet muliggjør.

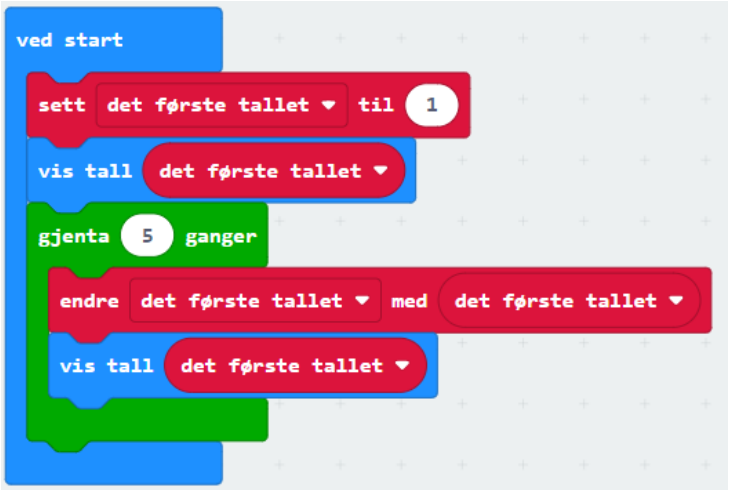
Ifølge Duval (2006, s. 111) kan et representasjonssystem kategoriseres som et register hvis systemet tillater veksling mellom representasjoner. Blokkprogrammering tillatter veksling mellom representasjoner, både som behandling og omdanning. Figur 2.10 viser eksempel på behandling av representasjoner for ulike tallfølger som øker med 1 og/eller 2. Heltallsfølgen {1, 2, 3, 4, 5, ...} gis som et resultat av programmet om man konsekvent trykker på knapp A. Heltallsfølgen er et matematisk objekt som representeres i begge programeksemplene, ved trykk på knapp A. Vi kan tenke oss at elever kan bli kjent med programmet i 2.10a og deretter går videre til å lage programmet vist i 2.10b, som representerer akkurat de samme tallfølgene. Programmet i 2.10b benytter betingelse for knappetrykkene, men innholdet i hvert knappetrykk og startverdien forblir det samme, noe som ikke påvirker tallfølgene den representerer. Begge programmene er en del av blokkprogrammering som system eller register. Vekslingen fra programmet i 2.10a til programmet i 2.10b blir derfor en behandling av representasjoner for det samme matematiske objektet.

(a) Program med separate knappe-trykk

(b) Program med betingelse

Figur 2.10: To representasjoner for de samme tallfølgene

Et eksempel på omdanning i blokkprogrammering kan vi se i figur 2.11. Dette programmet representerer det som i det symbolske registeret er tallfølgen {1, 2, 4, 8, 16, 32, ...}. Elever som kan lage dette programmet i MakeCode etter å ha studert tallfølgen, gjør en omdanning fra symbolsk register til blokkprogrammering som register.



Figur 2.11: Program for tallfølgen {1, 2, 4, 8, 16, 32, ...}

Omdanning krever forståelse for hvordan det matematiske objektet opptrer ulikt i de forskjellige registrene (Bråting & Kilhamn, 2021, s. 181). Bråting & Kilhamn (2021) bruker variabelbegrepet som eksempel. Hvis en elev beskriver en tallfølge rekursivt, for eksempel «tallfølgen starter med 1 og øker alltid med det forrige tallet» kan eleven innse behovet for en variabel i utformingen av programmet. Å innse behovet for variabel er nødvendig

for å lykkes med omdanningen fra naturlig språk til blokkprogram for tallfølger. Bråting og Kilhamn (2021) skriver at det er vesentlig for eleven å vite hvor variabelen skal plasseres, og hvilken funksjon variabelen har med hensyn til det matematiske objektet, for å kunne bruke variabelen som tegn i en semiotisk representasjon.

Bråting & Kilhamn (2021) benytter Duvals teori om registre i sitt arbeid med å sammenligne algebra og programmering. De skriver at ulike representasjonsregistre uttrykker ulike egenskaper ved det matematiske objektet som representeres (Bråting & Kilhamn, 2021, s. 174). For eksempel vil en graf av tallfølgen {1, 2, 4, 8, 16, 32, ...} visuelt beskrive tallfølgens systematiske økning. Et blokkprogram vil på sin side uttrykke økningen direkte, tilsvarende en funksjon. Blokkprogrammet vist i figur 2.11 benytter variabelblokken «endre (det første tallet) med (det første tallet)», hvor «det første tallet» har samme rolle som *n* i funksjonsuttrykket $a_n = a_{n-1} + a_{n-1}$. Her blir økningen uttrykt direkte, på en rekursiv måte. Endreblokken «endre (det første tallet) med (det første tallet)» gir assosiasjoner til det tilsvarende funksjonsuttrykket for tallfølgen, men blokkprogrammering skiller seg fra funksjonsuttrykk, fordi det bruker ord. Et funksjonsuttrykk skrives med symboler.

Blokkprogrammering har ulike fordelaktige egenskaper som representasjonsregister i elevers arbeid med matematikk. Som beskrevet i 2.2 vil blokkprogrammering hindre elever i å gjøre syntaksfeil, fordi blokkenes former gjør at kun gyldige blokksammensetninger kan skapes. Blokkenes former legger til rette for at elever alltid skaper program som fungerer. Programmet de lager behøver imidlertid ikke å bli slik de ønsket. Selv om programmene alltid vil utføre en handling, vil det ikke nødvendigvis være handlingen elevene trodde skulle skje.

En annen egenskap ved blokkprogrammering er hvordan elevene ikke behøver å kjenne til alle reglene i registeret for å lage representasjoner i det. I algebra er det nødvendig å kjenne til reglene for sammensetning av symboler for å lage gyldige påstander. Elever som ikke kjenner til algebraisk notasjon, kan heller ikke være sikker på om representasjonen de lager er gyldig. I blokkprogrammering kan man oppnå gyldige representasjoner av et matematisk objekt, gjennom prøving og feiling, fordi simulatoren eller micro:bit'n, kontinuerlig gir eleven tilbakemelding om hva programmet deres gjør. Hvis ikke programmet gjør det man ønsker kan man enkelt gjøre endringer og prøve ut nye løsninger.

2.5 Tidligere forskning på programmering for læring av matematikk

For å presentere tidligere forskning som er relevant for denne studien vil jeg innledningsvis presentere en litteraturstudie av Forsström & Kaufmann (2018). Deres studie er en gjennomgang av alle publiserte artikler fra perioden 1995-2018 som omhandler programmering i matematikkundervisning i grunnskolen. Videre vil jeg supplere med utvalgte forskningsartikler fra perioden 2019-2022 som alle har relevans for min studie. Jeg anvender Duvals (2006) teoretiske rammeverk om representasjonsregistre for å få innsikt i hvordan blokkprogrammering kan fungere som et semiotisk register for tallfølger. Basert på mine litteratursøk finnes det per i dag ingen tidligere forskning som har gjort dette. Riktignok har Bråting & Kilhamn (2021) tatt utgangspunkt i Duvals (2006) rammeverk om register, for å sammenligne programmeringsspråk og algebraisk notasjon som to ulike registre. Jeg

vil presentere artikkelen til Bråting & Kilhamn (2021), samt supplere med artikler som har brukt MakeCode som programmeringsmiljø i sine studier.

Forsströms & Kaufmanns (2018) hensikt var å forstå det vitenskapelige grunnlaget for at programmering har blitt plassert i matematikkfaget. Til tross for at de ønsker å forstå bakgrunnen for programmering i matematikkfaget, påpeker de at programmering ikke er et nytt fenomen i matematikkundervisning (Forsström & Kaufmann, 2018, s. 19). Papert (1980 i Forsström & Kaufmann, 2018) var tidlig ute med å skape et programmeringsmiljø som skulle bidra til læring av matematikk i sammenheng med programmering. Logo, som programmet het, baserte seg på at elever styrte en skilpadde på en dataskjerm, ved hjelp av programmering (Forsström & Kaufmann, 2018).

I 1995 ble det publisert forskning på Logos potensielle læringsutbytte i matematikk (Yelland, 1995 i Forsström & Kaufmann, 2018). Forskningen konkluderte med varierende grad av utbytte når det gjaldt problemløsning og matematisk oppnåelse (Yelland, 1995 i Forsström & Kaufmann, 2018). Forsström og Kaufmann (2018) forsøker med sin litteraturstudie å gi en oversikt over litteratur på programmering i matematikkundervisning fra 1995 og fram til 2018. Etter å ha samlet inn alle studier fra perioden som inkluderte matematikkundervisning, deltakere i grunnskolealder, programmering, og i noen tilfeller roboter, stod de igjen med 15 artikler (Forsström & Kaufmann, 2018). Ingen av artiklene tok i bruk micro:bit eller MakeCode som programmeringsmiljø. Scratch var imidlertid programmeringsmiljøet ved 5 av 15 studier.

Litteraturstudien til Forsström og Kaufmann (2018) identifiserte tre temaer som oppsummerte fokusområdene i de utvalgte studiene: motivasjon for matematikk, elevers prestasjon i matematikk og samarbeidet mellom elever og læreres nye rolle. Enkelte studier viste økt matematisk motivasjon ved bruk av programmering, men samlet sett var ikke resultatene generaliserbare (Forsström & Kaufmann, 2018, s. 26). Programmering ga økt matematisk prestasjon blant noen elevgrupper i alle studiene som målte prestasjon (Forsström & Kaufmann, 2018, s. 26).

De fleste av de 15 studiene baserte seg på et konstruktivistisk læringssyn (Benitti & Spolaôr, 2017 i Forsström & Kaufmann, 2018), slik også min studie gjør. For å best kunne vurdere programmeringens didaktiske potensiale i matematikkfaget mener Forsström og Kaufmann (2018, s. 30) at det er relevant å studere elevers kollektive læringsprosess i kontrast til individfokusert læringsutbytte.

Forsström og Kaufmann (2018, s. 30) konkluderer med at geometri benyttes av flere i sammenheng med programmeringsaktiviteter, men at det behøver mer forskning relatert til andre matematisk tema. Min studie omhandler tallfølger som mønster. Mønster i programmeringssammenheng har blitt undersøkt i senere tid av Miller (2019). Miller (2019) undersøkte hvorvidt programmering kan relatere til elevers identifisering av mønster og strukturer. Mønsteraktiviteter er hovedtemaet i Millers (2019) studie. I min studie fungerer mønster mer som et bakteppe for programmeringsaktivitetene. Miller (2019) fant at matematikklæring gjennom programmering kan føre til økt matematisk forståelse når det gjelder identifisering av mønster og strukturer.

Forsström og Kaufmann (2018) kommenterer at forskning på det didaktiske potensiale ved programmering i matematikkundervisning både er spredt og mangelfullt. I senere tid har

det tilkommet ny forskning som studerer hvordan programmering kan forenes med matematikkfaget, men jeg vil hevde at forskningsfeltet fortsatt er spredt og mangelfullt. Av litteraturen jeg har funnet som innfrir Forsströms og Kaufmanns (2018) kriterier, er det sjelden at flere artikler behandler det samme matematiske tema. Programmering har blitt studert i sammenheng med matematisk argumentasjon av Kaufmann og Stenseth (2021). Laurent et al. (2022) gjorde en kvantitativ studie av elevers matematiske prestasjon ved blokkprogrammering relatert til divisjon, brøk og funksjoner i Scratch. Benton et al. (2018) undersøkte hvorvidt programmering i Scratch kunne fasilitere elevers læring av titallssystemet, gjennom «ScratchMaths», et toårig intervensjonsprosjekt som forsøker å knytte programmering sammen med matematisk læringsutbytte. Ifølge Benton et al. (2018) er det mulig å oppnå en forening av programmering og matematikk i Scratch, så lenge læreren gir tilstrekkelig støtte, og undervisningen er forankret i et didaktisk grundig undervisningsopplegg.

Bråting & Kilhamn (2021) og Kilhamn et al. (2022) relaterer programmering til algebra. Begge disse artiklene bruker Duvals (2006) rammeverk om representasjonsregistre for å sammenligne algebra og programmering. Jeg vil nå presentere dem da de er særlig relevante for min studie.

Bråting & Kilhamn (2021) gjorde en empirisk studie som undersøkte sammenhenger mellom algebraisk tenkning og programmering i matematikkundervisning, med utgangspunkt i Duvals (2006) teori om semiotiske representasjoner. Et bakteppe for diskusjon er hvordan aspekter ved algoritmisk tenkning kan utvikles gjennom programmering (Bråting & Kilhamn, 2021). De brukte to forskningsspørsmål for å undersøke sammenhengen mellom matematikk og programmering. Det første omhandlet likheter og ulikheter i hvordan begrepene variabel, likhet og funksjoner representeres i programmeringsspråk og algebraisk notasjon. Det andre undersøkte hvordan innføringen av programmering i skolematematikken potensielt kan gi muligheter og utfordringer med tanke på utviklingen av algebraisk tenkning (Bråting & Kilhamn, 2021). Studiens empiri baserer seg på tre ulike programmeringsaktiviter, gjort på tre ulike aldersgrupper. Både blokkbasert og tekstbasert programmering er inkludert i studien. Hovedargumentet i studien er at det er forskjeller mellom det symbolske registeret og programmering som register, og at dette må tas hensyn til med tanke på elevers arbeid med omdanning. De understreker at forskjeller og likheter mellom ulike registre kan skape muligheter såvel som utfordringer i elevenes arbeid med registrene (Bråting & Kilhamn, 2021).

Kilhamn et al. (2022) har publisert en studie som undersøker hvordan blokkbasert programmering gjennom Scratch kan ha et didaktisk potensial i matematikkundervisning i sammenheng med algebraisk tenkning. Kilhamn et al. (2022) sin empiriske studie har likheter med Bråting & Kilhamn (2021), ved at begge betrakter algoritmisk tenkning og benytter Duvals (2006) perspektiver på semiotiske representasjoner. I Kilhamn et al. (2022) sin studie vektlegges variabelkonseptet, med utgangspunkt i at variabler må forstås ulikt i algebra, som del av det symbolske registeret, og i programmering som et annet register. Et sentralt poeng i studien er hvordan programmering i tidlige skolealder kan tilføre nye måter å betrakte variabelkonseptet, ved å gjøre registerskifte mellom naturlig språk eller symbolsk register og programmering som register (Kilhamn et al., 2022, s. 1286).

Jeg har kun funnet to artikler som knytter sammen programmering med matematikk i

programmeringsmiljøet MakeCode. Andersen (2022) har gjennomført en toårig kvalitativ studie som undersøker på hvilke måter blokkprogrammering kan forenes med matematikkfaget. Andersen (2022) tar også for seg algoritmisk tenkning som en del av studien. Gjennom en rekke intervensjoner studerte hun elevers programmering i MakeCode og micro:bit relatert til sannsynlighet som matematisk tema. Andersen (2022, s. 18) konkluderte med tre funn; blokkprogrammering i matematikk muliggjør samarbeidslæring, programmering i matematikk muliggjør utvikling av algoritmisk tenkning og blokkprogrammering fasiliterer læring av matematikk.

Kazi (2022) har også undersøkt programmering knyttet til matematikk ved bruk av MakeCode og roboten Finch. Artikkelen er skrevet av en doktorgradsstudent og er tilsynelatende ikke fagfellevurdert. I hans studie er MakeCode programmeringsmiljøet som lager program til roboten Finch. Kazi (2022) kombinerer bruk av kvantitative og kvalitative forskningsmetoder og undersøker hvordan programmering med roboten Finch utvikler elevers mestringsforventning og presentasjon i matematikk. De kvantitative funnene viste at programmeringen hadde negativ effekt på elevenes mestringsforventning og ingen signifikant effekt på prestasjon i matematikk. De kvalitative funnene viste positiv effekt på begge områdene (Kazi, 2022). Samlet sett tyder funnene på at elevenes mestringsforventning hverken har signifikant positiv eller signifikant negativ påvirkning på elevenes matematisk prestasjon (Kazi, 2022).

3 Metode

I denne studien undersøker jeg *hvordan elever på 4. trinn bruker blokkprogrammering som semiotisk representasjonsregister i arbeid med tallfølger*. Studien har en kvalitativ tilnærming, fordi jeg forsker på mennesker og deres verbale og skriftlige samhandling i oppgaveløsning (Clark et al., 2021). Kvalitative forskning har et dybdeperspektiv på menneskers handlinger, hvor disse handlingene gir grunnlag for nye betraktninger på forskningsområdet (Clark et al., 2021, s. 350). Mitt forskningsprosjektet er gjennomført med en designbasert kontekst, hvor iterasjonen mellom teori og undersøkelse vektlegges (Prediger et al., 2015). Gjennom konstant komparativ analysemetode har jeg vekslet mellom å studere datamaterialet og Duvals (2006) teori om semiotiske representasjonsregistre i samsvar med datamaterialet. I analyseprosessen har jeg dermed hatt en abduktiv tilnærming ved å først analysere datamaterialet med åpen koding og deretter knyttet kodene til Duvals (2006) teorier.

For å undersøke elevers arbeid i matematikk tok jeg i bruk observasjon som forskningsmetode, både i form av video- og lydopptak, og skjermopptak fra elevenes programmeringsprosess. For å svare på problemstillingen ut ifra et konstruktivistisk læringssyn, benyttet jeg undervisningsmodellen PRIMM som utgangspunkt for undervisningsopplegget. PRIMM er spesifikt utviklet for programmeringsundervisning med fokus på samarbeid og dialog (Sentance et al., 2019). Elevene som deltok i denne studien har en relasjon til meg som lærerstudent i praksis. Datainnsamlingen foregikk i praksisperioden og ukene etter.

3.1 Forskningsdesign

Problemstillingen for denne studien undersøkes i en designbasert kontekst, også kalt intervensjonsstudie, designstudie eller «developmental research» (Prediger et al., 2015; van den Akker et al., 2006). Designforskning kan anses som et samlebegrep for flere relaterte forskningsdesign som deler samme hensikt og/eller karakteristikker (van den Akker et al., 2006). Jeg baserer min studies forskningsdesign på Prediger et al. (2015) sine betraktninger om designbasert forskning. Prediger et al. (2015) benytter begrepet «design research». Jeg kaller derfor min studie for en designstudie.

Van den Akker et al. (2006) påpeker at den viktigste årsaken til å benytte designforskning er, og har vært, å forsterke forskningens betydning i utdanningssystemet. Designbasert forskning som forskningsdesign får økende oppmerksomhet som betydningsfull metodologi i matematikkdidaktisk forskning (Prediger et al., 2015). Designforskning kan skisseres ved hjelp av fem karakteristikker (Prediger et al., 2015; van den Akker et al., 2006). Jeg vil nå beskrive de ulike karakteristikkene og eksemplifisere hvordan denne studien samsvarer med dem. Beskrivelsene er hentet fra Prediger et al. (2015, s. 879).

1. *Intervensjon:* designforskning skal undersøke og utvikle nye praksiser.

   Denne studien undersøker programmering som representasjonsregister i en klassekontekst hvor læreren og skolen ønsket mer kunnskap om programmering.

2. *Generere teori:* designforskning skal videreutvikle etablert teori eller danne ny teori med utgangspunkt i elevenes læringsprosesser.

Duvals (2006) teori om semiotiske representasjonsregistre inkluderer ikke programmering. Jeg ønsker å anvende Duvals etablerte teori for å undersøke hvordan blokkprogrammering kan forstås ut ifra hans rammeverk. Denne innovasjonen kaster nytt lys over en etablert teori.

3. *Refleksiv prosess:* designforskning ser sammenhengen mellom teori og undersøkelse. Teori kan styre hvordan vi utformer et undervisningsopplegg eller undersøkelse, men prosessen med å analysere undersøkelsen, og å gjøre tilpasninger i den, kan også anses som en videreutvikling av teorien.

   I denne studien utformet jeg et undervisningsopplegget for å undersøke elevers bruk av ulike representasjonsregistre ved programmering. Etter første gjennomføring strevde elevene med omdanning, noe som medførte justeringer i undervisningopplegget. Justeringene vil bli diskutert som et ledd i å forstå mer om blokkprogrammeringens rolle i Duvals (2006) teori.

4. *Iterasjon:* designforskning er en syklisk prosess hvor man veksler mellom antatt resultat av undersøkelsen og det faktiske resultatet av undersøkelsen. Kunnskap utvikles ikke bare ved iterasjon, men i analysen av iterasjonen.

   Jeg gjennomførte undervisningsopplegget i denne studien to ganger, og gjorde noen justeringer fra første til andre gjennomføring. En iterativ prosess er for øvrig en naturlig del av praksisorientert forskning, fordi vi aldri kan forutse elevers tanker og svar fullstendig. Som lærere gjør vi alltid tilpasninger underveis og til neste gang.

5. *Praksisorientert gyldighet:* designforskning må ha en praksisnær validitet eller gyldighet. Gravemeijer & Cobb (2006) argumenterer for at intervensjoner i designforskning ikke kan være fullstendig replikerbare, men at resultatet av forskningen skal danne et grunnlag for å anvende funnene til andre situasjoner. For å oppnå en praksisorientert gyldighet er det avgjørende med en grundig beskrivelse av elevene, deres læringskontekst, gjennomføringen av undersøkelsen og ikke minst hvordan disse faktorene påvirker læringsprosessen (Prediger et al., 2015). Basert på situasjonsbeskrivelsen som er gitt skal nye lærere eller forskere kunne relatere studien til sitt eget arbeid (Gravemeijer & Cobb, 2006).

   Denne studien forsøker å gi en grundig situasjonsbeskrivelse av både elevene, deres tidligere programmeringsundervisning, forløpet til undervisningsopplegget, og selve gjennomføring av undervisningen.

3.2 Beskrivelse av utvalg og kontekst

Valg av skole og trinn er ikke tilfeldig i denne studien. Som beskrevet innledningsvis, er denne masteroppgaven en del av forskningsprosjektet LAB-TEd. Siden jeg deltar i dette prosjektet, er min datainnsamling betinget i min praksisperiode og elevene til min praksislærer. Elevene som deltar i denne masteroppgaven er 9 år i forskningsperioden og går i 4. klasse på en skole i Trøndelag. Elevene kjente studentgruppa fra en tidligere praksisperiode på høsten. Høstpraksisen varte i to uker, hvor programmering var fokus for alle matematikkøktene. Vi i studentgruppa hadde i samarbeid med veiledere fra universitetet og praksislærer allerede etablert at programmering var overordnet tema for våre masteroppgaver. Vi bestemte oss derfor for å planlegge programmeringsundervisning for hele praksisperioden.

Vi begynte med analog programmering for å introdusere elevene for programmering. Alle aktivitetene vi gjennomførte gikk ut på at læringspar samarbeidet om å utføre en oppgave, hvor den ene eleven kun forklarte og den andre kun utførte handlingen. Hensikten med aktivitetene var å tydeliggjøre behovet for presis kommunikasjon. En helthetlig oversikt over perioden jeg var sammen med elevene er gitt i tabell 3.1. Datainnsamlingen i dette prosjektet foregikk i uke 2 og uke 4.

Tabell 3.1: Tidslinje

Uke 39-40	Uke 1	Uke 2	Uke 4
Praksis høst	Introduksjon til programmeringsvariabel	Praksis vår	Datainnsamling

I løpet av praksisperioden satte vi også av tid til å gi elevene en introduksjon i MakeCode og grunnleggende programmeringsblokker, blant annet løkker, «vis tall» og knapper som inndata. Elevene fikk i begynnelsen muligheten til å utforske programmet, og programmering generelt, på egen hånd. Vi lagde oppgaver som elevene kunne prøve seg på, men ga ingen instruksjon eller støtte i form av modellering i forkant. Senere fikk elevene også anledning til å analysere program vi hadde lagd på forhånd. Se vedlegg 1. Elevene fikk også teste micro:bit som fysisk objekt i løpet av disse ukene.

I etterkant av vår praksisperiode på høsten hadde elevene noen undervisningsøkter i Scratch. Vi kom tilbake i uke 2 og gjennomførte vår siste praksisperiode. Med utgangspunkt i vår forskning, som foregikk samme uke og ukene etter, planla vi matematikkundervisning med programmering og MakeCode som fokus. På dette tidspunktet hadde elevene noe kjennskap til programmering og MakeCode fra høstpraksisen, men det var noen måneder siden de hadde brukt det sist. Tatt i betraktning elevenes pause fra MakeCode, og manglende erfaring med variabel i programmering, planla vi en programmeringstime fredagen før praksisuka som matematikklæreren på trinnet gjennomførte. Se uke 1 i tabell 3.1. Undervisningen var en introduksjon til variabelbegrepet i programmering. Aktivitetene foregikk på MakeCode. Vi planla denne økta, fordi vi visste at elevene behøvde noe kunnskap om variabelbegrepet for å kunne gjennomføre aktivitetene vi ønsket å bruke i våre forskningsprosjekter. Samtidig ga undervisningen en mulighet til å observere hva elevene mestret i aktivitetene.

Totalt har 7 elever deltatt i dette prosjektet, 4 elever på den første gjennomføringen og 3 elever på andre gjennomføring. I den første gjennomføringen valgte jeg elever som allerede var etablerte læringspar, altså to og to. I den andre gjennomføringen tok jeg ikke hensyn til etablerte læringspar, men de 3 valgte elevene ble organisert som et læringspar og en elev som arbeidet selvstendig. Av praktiske årsaker valgte jeg kun 3 elever i den andre gjennomføringen. I utgangspunktet skulle de også være 4 elever. Jeg valgte elever til studien ut ifra tre betingelser: de samtykket til prosjektet, de fulgte ordinær undervisning og de hadde lite kjennskap til programmering. Jeg la vekt på disse tre betingelsene, for å velge elever med «typiske» forutsetninger for å mestre oppgavene. Hensikten med utvalget var å velge elever som representerer en typisk 4. trinnselev. Datainnsamlingen foregikk i elevenes matematikkøkter.

3.3 PRIMM

Jeg benytter PRIMM-modellen som utgangspunkt for undervisningsopplegget i studien. I prosessen med å utforme undervisningsopplegget undersøkte jeg tidligere studier som omhandlet programmeringsundervisning. Sentance et al. (2019) har utviklet PRIMM, et rammeverk for programmeringsundervisning. PRIMM er et nyutviklet, forskningsbasert rammeverk for strukturert undervisning av programmering (Sentance et al., 2019). Utviklingen av PRIMM knyttes ikke spesifikt til matematikkundervisning, men snarere til programmeringsundervisning uavhengig av fag og læreplan.

Diskusjoner om didaktiske og kognitive utfordringer relatert til programmeringsundervisning førte til utviklingen av PRIMM (Sentance et al., 2019). Grad av struktur i undervisningen, og programlesing i motsetning til programskriving (programmering), var to momenter som ble diskutert av forskerne (Sentance et al., 2019). PRIMMs strukturerte tilnærming gir lærere anledning til å støtte elevers forståelse av essensielle programmeringskonsepter (Sentance et al., 2019). Den strukturerte tilnærmingen støtter elevenes utvikling av konsepter ved programmering gjennom fokus på følgende områder:

- Programlesing før programskriving
- Samarbeidsbasert læring, med vekt på språklig samhandling
- Undersøkelse av programmets enkeltdeler, for deretter å oppnå forståelse for hele programmet
- Gradvis overgang fra andres program til eget program; eierskap og autonomi

Sentance et al. (2019, s. 15-16, egen oversettelse)

PRIMM består av de faste stegene «predict», «run», «investigate», «modify» og «make». Jeg vil nå gjengi deres innholdet, basert på Sentance et al. (2019, s. 13-15).

«**Predict**» (forutse)
Elevene blir presentert for et program på papir. Elevene leser programmet og danner seg en mening om hva programmet vil gjøre. Antagelsen kan skrives ned eller deles muntlig. Læreren diskuterer antagelsene med klassen.

«**Run**» (kjøre)
Programmet kjøres, uten at elevene får anledning til å bygge opp programmet manuelt. Elevene i denne studien fikk utdelt en URL de skulle gå innpå, se 3.4.1 Første økt.

«**Investigate**» (utforske)
Elevene observerer hvorvidt deres antagelser om programmet stemmer. Læreren stiller spørsmål relatert til elevenes programforståelse. Hvorfor behøver vi denne blokka? Hva gjør «endre» med variabelen? Spørsmålene kan være relatert til enkeltdeler eller hele programmet. Diskusjonen omkring spørsmålene bør ideelt sett foregå i par eller grupper, slik at elevene får anledning til å utvikle et språk for å snakke om programmering og programmet som inngår i undervisningen (Cutts et al., 2012 i Sentance et al., 2019).

«**Modify**» (modifisere)

Elevene har nå kjennskap til, eller forståelse av, programmet de har jobbet med. De blir bedt om å gjøre forandringer i programmet. Enkle forandringer i begynnelsen, deretter mer betydningsfulle forandringer. Eksempelvis: Kan du endre programmet slik at det første tallet blir 5? (Det første tallet var originalt 1, ergo kreves en endring av variabelens startverdi). Forkunnskapen elevene har fra det originale programmet skal gi elevene nødvendig selvtillit for å videreutvikle programmet.

«**Make**» (skape)
I det siste steget skal elevene selv skape et nytt program basert på kunnskapen ervervet fra de tidligere stegene. Det nye programmet skal utvikles basert på et problem eller kontekst gitt av læreren. I denne studien ga jeg elevene en helt ny tallfølge for skape-steget. Denne tallfølgen var rammen for programmet de skulle skape i læringspar.

Skaperne av PRIMM presiserer at stegene ikke behøver å gjøres i den faste rekkefølgen, men at det kan være behov for å gjenta enkelte steg i modellen før man går videre til et nytt steg (Sentance et al., 2019). Undervisningsopplegget i denne studien følger stegene slik: P-R-I-M-I-M. Med andre ord gjennomførte vi en ekstra utforskingsfase før vi gikk over til siste steg. Et undervisningsopplegg med PRIMM behøver heller ikke å foregå kun som én skoletime (Sentance et al., 2019). Mitt undervisningsopplegg ble gjennomført over to økter, hver med 90 minutter avsatt.

3.4 Undervisningsopplegget

Undervisningsopplegget i min studie er fordelt over to økter. Totalt sett følger undervisningsopplegget PRIMM-modellen på denne måten: P-R-I-M-I-M, hvor P-R-I-M foregikk i den første økta og I-M foregikk i den andre økta. Den andre økta begynte med en repetisjon hvor jeg og elevene snakket om blokkprogrammet fra den første økta, før vi gikk over til programmering. Den andre økta består derfor av både I og M, utforsking og skaping. Utforskingen i begynnelsen av andre økt hadde som hensikt å minne elevene på hvordan programmet så ut forrige gang, og gjenta hva de ulike blokkene i programmet betydde, slik at elevene fikk en enklere overgang til skape-fasen, hvor de måtte konstruere et program på egenhånd.

Jeg organiserte elevene i faste læringspar. Hvert par arbeidet sammen på én Chromebook. I andre gjennomføring av undervisningsopplegget var det kun tre elever. For denne gruppen lagde jeg et læringspar og en som arbeidet selvstendig. Hensikten med å la læringspar samarbeide om en Chromebook var å tilrettelegge for at læringsparet tok i betraktning hverandres ideer med programmet.

I alle gjennomføringene plasserte jeg elevene ved siden av eller ovenfor hverandre, slik at de kunne høre og snakke med hverandre. Hensikten med dette grepet var å legge til rette for dialog. I denne studien er jeg interessert i elevenes samtale, fordi det gir meg mulighet til å vurdere om elevene gjør omdanning mellom blokkprogrammering og naturlig språk som representasjonsregister.

I undervisningsoppleggets første økt samarbeidet også elevene om samme oppgaveark. Eleven som arbeidet alene hadde sitt eget oppgaveark, men kunne likevel diskutere med de to andre elevene. I den andre økta fikk elevene hvert sitt oppgaveark. Hensikten med å få eget ark var at elevene skulle få mulighet til å forklare tallfølgen på sin egen måte og

i sitt eget tempo, før de etterpå skulle samarbeide om å skape denne tallfølgen i programmeringsmiljøet.

3.4.1 Første økt

Den første økta kan sees i sin helhet på figur 3.1 eller vedlegg 3. For å inkorporere tallfølger med programmering er den første oppgaven i undervisningsopplegget å studere en tallfølge, kommentere hva som blir de neste tre tallene, og vise hvorfor. Denne oppgaven har ikke direkte sammenheng med PRIMM, men inkluderes ut fra denne studiens formål. Deretter fikk elevene se et program i papirformat som de skulle analysere og forklare hvordan fungerte. Når alle hadde skrevet eller sagt hva de mente programmet ville gjøre, fikk elevene lenken til programmet. Lenken tok elevene direkte inn i MakeCode, hvor programmet fra oppgavearket allerede var ferdiglagd av meg. Å gi elevene det ferdiglagde programmet er et poeng i PRIMM, fordi de anser manuell oppbygging av et program som en annen type prosess (Sentance et al., 2019, s. 13).

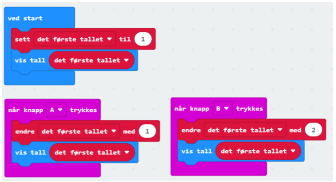
Når elevene kom seg inn på lenken med Chromebook fikk de trykke på knappene og prøve seg fram slik de selv ville for å utforske hvordan programmet faktisk fungerte. Jeg stilte elevene spørsmål underveis i utforskingen for å prøve å finne ut hvordan de forsto de ulike blokkene i programmet. For eksempel: hvorfor behøver vi den blå «ved start»-blokken? Dette steget er I i PRIMM, utforskingssteget. Etter utforsking og spørsmål fra meg fikk elevene i oppgave å endre programmet, i tråd med modifiseringssteget i PRIMM. Elevene skulle først endre programmet slik at det første tallet ble 5. Deretter skulle de anta hvilken tallfølge som kom til å vises hvis de trykket A, B, A, B,.. . Deretter skulle de notere ned hva skjermen faktisk viste. Elevene fikk ytterligere to oppgaver som omhandlet modifisering. Begge oppgavene inneholdt nye tallfølger, men som begynte med 5. Den første oppgaven innebar og forandre rekkefølgen man trykker på knappene på. Den andre, og siste oppgaven, krevde at elevene la til et ekstra knappetrykk (A+B) i programmet for å vise den nye tallfølgen på skjermen. Se oppgave 5 c i figur 3.1b. Denne første økta hadde som hensikt å fungere som en introduksjon for å gå videre til det selvstendige arbeidet i den andre økta. Den andre økta, som tar for seg PRIMMs siste steg, er økta som denne studien hovedsakelig tar utgangspunkt i. Det er denne økta som i all hovedsak relaterer til problemstillingen i min studie.

Mandag 09.01.23

10.30-12.00

1. Se på denne tallfølgen: 1, 2, 4, 5, 7, 8 … Hva blir de tre neste tallene i følgen? Forklar eller vis hvorfor.

2. Se på koden under. Skriv hva du tror den gjør. Hva vil vises på Micro:bit'n sin skjerm?



1

3. Kjør koden som dere får som lenke. Prøv å trykk på de ulike knappene og se hva som skjer.

4. Skjedde det som du forventet? Viste skjermen det samme som du trodde?

5. Nå skal du få endre på koden selv.

   a. Endre koden slik at det første tallet i tallfølgen blir 5.

Hva tror du tallfølgen blir om du nå trykker A, B, A, B…?

Hva viste Micro:bit'n når du trykket A, B, A, B…?

   b. Endre koden slik at du ved hjelp av knappene får tallfølgen 5, 6, 8, 10, 11, 13…

   c. Endre koden slik at du ved hjelp av knappene får tallfølgen 5, 7, 8, 13, 15, 16, 21..

   d. Endre koden slik at du får en tallfølge som du selv vil ha.

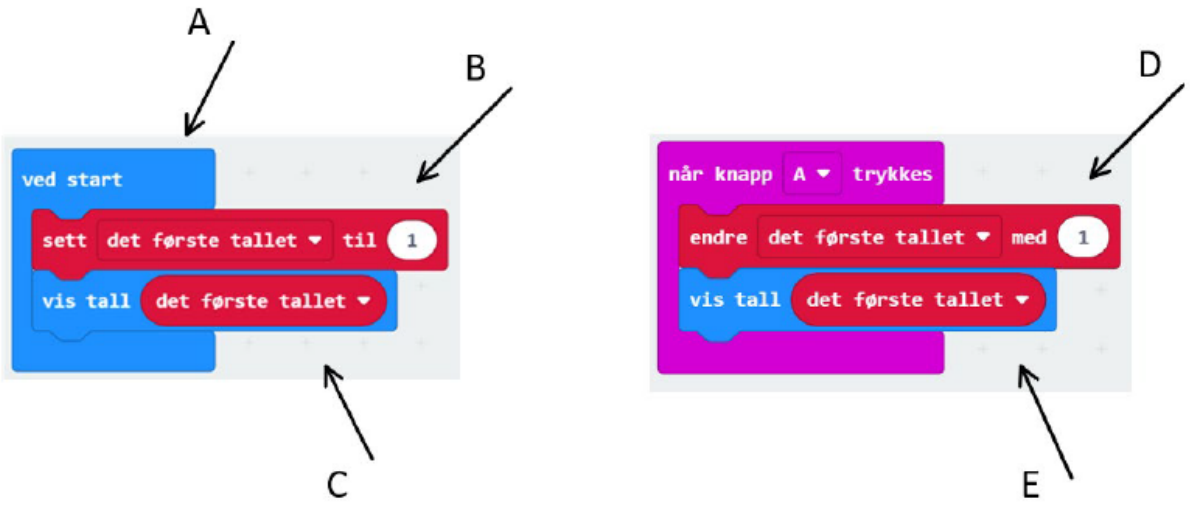
2

(a) side 1 (b) side 2

Figur 3.1: Oppgaveark første økt

3.4.2 Andre økt

Som nevnt besto den andre økta av utforsking (I) og skaping (M). Repetisjonen, eller utforskingen, gikk ut på at jeg spurte elevene om hver enkelt blokk fra forrige økt og hva blokkene betydde (se figur 3.2). Elevene fikk også utdelt et sett med setninger som forklarte hver blokk. (Se figur 3.2 sammen med 3.3). Før elevene fikk forklaringslappene vist i figur 3.3, fjernet jeg bokstavene som avslører hvilken forklaring som tilhører hvilken pil. På denne måten måtte elevene selv matche riktig forklaring til riktig pil. Repetisjonen begynte i plenum, men når elevene fikk utdelt forklaringslapper plasserte jeg dem i læringspar. I andre gjennomføring av undervisningsopplegget var det bare 3 elever, her samarbeidet alle tre om forklaringslappene.



Figur 3.2: Programmet fra første økt, bortsett fra knapp B

A:	Informasjonen begynner med en gang koden kjøres
B:	Bestemmer en startverdi som senere kan endres
C:	Viser variabelen som ble satt i begynnelsen, på skjermen
D:	Endrer variabelen slik at den øker med en
E:	Viser variabelen etter den ble endret

Figur 3.3: Forklaringslapper til programmet

Etter repetisjonen fikk elevene følgende oppgave: «Det finnes en tallfølge {1, 2, 4, 8, 16, ...}. Lag et program som viser de 6 første tallene i denne tallfølgen, uten å bruke knapper». Jeg leste opp oppgaven høyt for elevene og de noterte tallfølgen på sine papir. Elevene arbeidet fortsatt i par og hvert par samarbeidet på én Chromebook. Elevene hadde tilgang til programmet vi brukte i første økt, i papirformat. De hadde derfor anledning til å kopiere noe av eller hele det gamle programmet i konstruksjonen av det nye. Programmet de skulle skape skulle ikke benytte micro:bit'n sine knapper, men likevel benytte variabel på liknende måter som programmet fra første økt, se figur 3.4a. Oppgaven ekskluderte bruk av knapper helt bevisst, fordi jeg anser bruk av knapp som en modifisering av det forrige programmet, ikke en skaping av et nytt program.

Undervisningsopplegget i denne studien tar for seg 5 ulike tallfølger, hvor jeg kategoriserer alle som voksende mønster, hvilket vil si at jeg forventer elevsvar hvor tallfølgen fortsetter å øke, ikke at den gjentar seg på nytt fra start som et repeterende mønster. Ved å på forhånd kategorisere tallfølgene som voksende mønster overfor elevene, unngikk elevene å tolke tallfølgen som repeterende mønster.

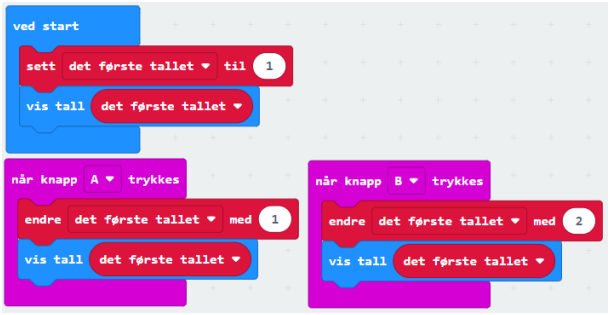
Tabell 3.2: Tallfølger brukt i denne studien

Tallfølge	Mønstertype (sekvensen det økes med)
{1, 2, 4, 5, 7, 8, ...}	Voksende (+1 +2)
{5, 6, 8, 9, 11, ...}	Voksende (+1 +2)
{5, 6, 8, 10, 11, 13, ...}	Voksende (+1 +2 +2)
{5, 7, 8, 13, 15, 16, 21, ...}	Voksende (+2 +1 +5)
{1, 2, 4, 8, 16, ...}	Voksende (+det forrige tallet)

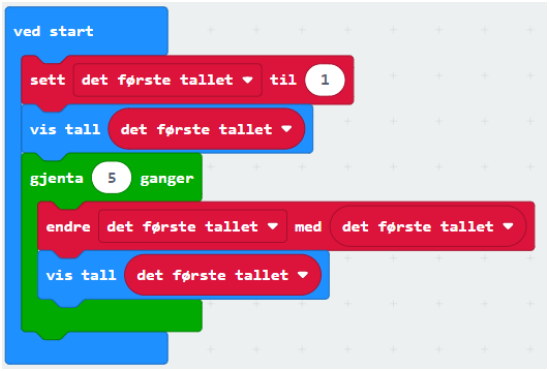
I tabell 3.2 fremgår alle tallfølgene som er brukt i denne studien. Den siste tallfølgen skiller seg ut fra de foregående. De foregående har varierende økning, men sekvensen har gjentatt seg likt hver gang. For eksempel sekvensen +1 +2 gjentar seg regelmessig i tallfølgen {1, 2, 4, 5, 7, 8, ...}. Tallfølgen {1, 2, 4, 8, 16, ...} har en økning som er systematisk, men ikke repeterende. Selve økningen øker for hvert nye tall. Overfor elevene og leseren av denne studien velger jeg å formulere økningen slik: det neste tallet i følgen er det forrige tallet summert med seg selv. Den neste økningen er alltid verdien til det forrige tallet.

3.4.3 Valg av blokkprogram

I arbeidet med å velge blokkprogram for den første økta (P-R-I-M) var det utfordrende å skape et program som ga tallfølger hvor programmet samtidig ikke ble for avansert med tanke på elevenes daværende programmeringsferdigheter. Bruk av variabel ble også nødvendig for å skape et program som ga tallfølger. Jeg endte med å skape programmet gjengitt i 3.4a, som kan veksle mellom mange ulike tallfølger med en økning på 1 og/eller 2. Ved bruk av flere inndata, for eksempel knapp «A+B», kan man legge til flere endringer av variabelen. Programmet tillater også endring av variabelens startverdi. Oppsummert gir programmet i 3.4a mange muligheter, samtidig som det er få ulike blokker involvert. Begge disse faktorene var avgjørende for at jeg valgte å benytte dette programmet i den første økta.



(a) Programmet brukt i første økt



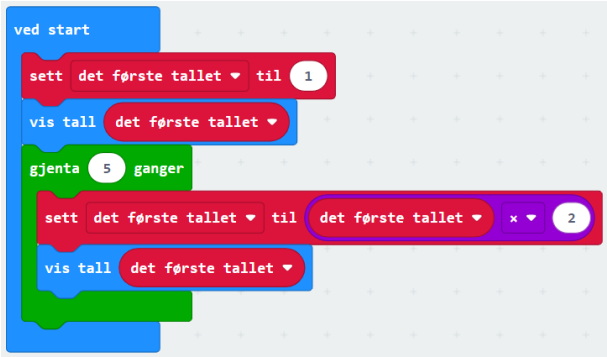
(b) Programmet brukt som utgangspunkt i andre økt

Figur 3.4: Programmene brukt i undervisningsopplegget

I den andre økta skulle elevene skape et program som ga tallfølgen {1, 2, 4, 8, 16, ...}. Elevene lagde mange egne program i forsøk på å oppnå fasiten. Fasiten er gitt i figur 3.4b. Jeg valgte dette programmet og dens respektive tallfølge som utgangspunkt for den andre økta, for å imøtekomme skape-steget i PRIMM. I skape-steget skal elevene skape

et program som inkluderer elementer fra det forrige programmet, uten at programmene er like (Sentance et al., 2019). Programmet i figur 3.4b inkluderer mange av de samme blokkene, men programmet krever en annen sammensetning av blokkene.

Noen studier rapporterer at barn i arbeid med tidlig algebra har en tendens til å generalisere mønster basert på rekursive beskrivelser, i motsetning til eksplisitte beskrivelser (Ferrara & Sinclair, 2016). I forbindelse med tallfølgen i den andre økta, {1, 2, 4, 8, 16, ...}, finnes det to mulige rekursive beskrivelser. Hver beskrivelse har et tilhørende program. Figur 3.4b og 3.5 viser to ulike program som fungerer for tallfølgen {1, 2, 4, 8, 16, ...}.



Figur 3.5: Annen mulig løsning for {1, 2, 4, 8, 16, ...}

Programmet i figur 3.4b hører sammen med mønsterforklaringen *det neste tallet i følgen er det forrige tallet summert med seg selv*. Dette er en rekursiv beskrivelse basert på en additiv økning. En annen rekursiv beskrivelse som baserer seg på multiplikativ økning er, *det neste tallet er det forrige tallet multiplisert med 2*. Denne forklaringen samsvarer med programmet i figur 3.5. Hvilken formulering man velger avgjør hvordan programmet må konstrueres i MakeCode.

Jeg valgte å fokusere på programmet i figur 3.4b når jeg veiledet elevene. Valget falt på dette programmet, fordi elevene hadde utforsket endreblokken i den første økta i undervisningsopplegget, slik vi kan se i figur 3.4a. Elevene hadde ikke tidligere prøvd å bruke to settblokker med samme navn. Min vurdering i planleggingsfasen var derfor at to settblokker med samme navn, altså å sette to startverdier som begge tilsynelatende er den samme, ville være mindre intuitivt enn å fortsette å bruke endreblokk, til tross for at variabelen inngår to ganger i samme blokk. Å fokusere på endreblokk, i stedet for settblokk, tilsvarer et rekursivt fokus, der endringen er basert på gjentatt addisjon.

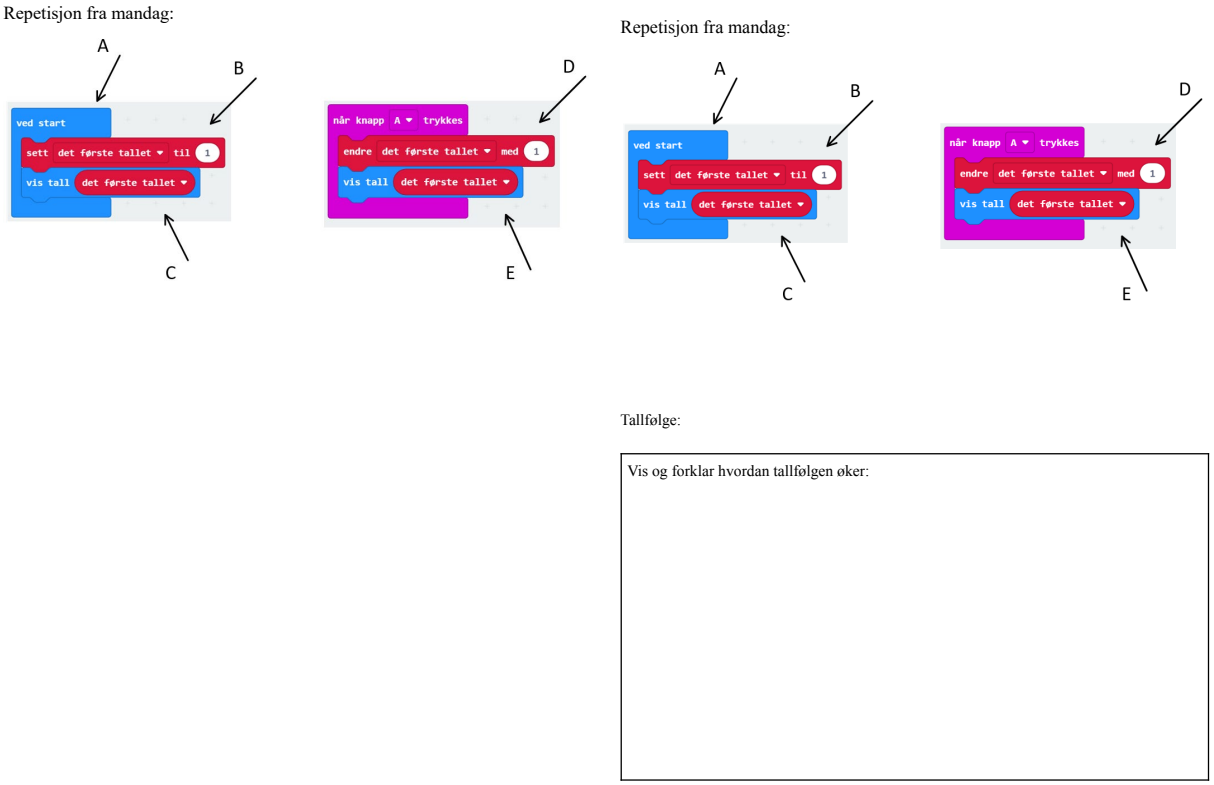
Blokken «endre (det første tallet) med (det første tallet)» i figur 3.4b indikerer at neste tall er en endring av det foreløpig tallet, med det foreløpig tallet. Altså, det neste tallet vil bli det tallet man har nå, økt med seg selv. Ved start har man tallet 1. For å kunne gi det neste tallet skal programmet endre tallet du har nå (1) med seg selv. Det vil gi tallet 2.

3.4.4 Endringer fra første til andre gjennomføring

Denne studien er en designstudie hvor iterasjon mellom antatt resultat og faktisk resultat er interessant (Prediger et al., 2015). I utforming av undervisningsopplegget hadde jeg en antakelse om hvordan elevene ville respondere på oppgavene. Etter første gjennomføring

erfarte jeg at det var behov for å gjøre tiltak i opplegget for å imøtekomme elevene bedre. Undervisningsopplegget, slik det er beskrevet, gjenspeiler den første gjennomføringen. Jeg vil nå kommentere hvilke endringer som ble gjort før den andre gjennomføringen.

Begge endringene jeg gjorde tilhører den andre økta. På figur 3.6 ser vi oppgavearkene fra første og andre gjennomføring av økt nummer to, ved siden av hverandre. Oppgavearkene kan også sees i vedlegg 4 og 5. I den andre gjennomføringen la jeg til en forklaringsboks hvor elevene skulle beskrive hvordan tallfølgen økte. Hensikten med dette tillegget var at elevene skulle bli mer bevisst på tallfølgens økning og hvilken forståelse de hadde for økningen. Additiv forståelse basert på gjentatt addisjon versus multiplikativ forståelse basert på multiplikasjon med 2.



(a) Oppgaveark fra første gjennomføring (b) Oppgaveark fra andre gjennomføring

Figur 3.6: Oppgaveark fra andre økt, begge gjennomføringene

Den andre endringen var at elevene skulle få lov til å bruke knapper hvis det viste seg at program uten knapper ble for utfordrende. Dette ble en løpende vurdering, men alle elevene i andre gjennomføring fikk etter hvert anledning til å bruke knapper.

3.5 Metode for datainnsamling

Min problemstilling er: *hvordan bruker elever på 4. trinn blokkprogrammering som semiotisk representasjonsregister i arbeid med tallfølger?* Forskningsobjektet i studien er elever og deres arbeid med programmeringsaktiviteter i matematikkfaget. For å kunne gi en nærgående beskrivelse av elevenes bruk av blokkprogrammering som representasjonsregis-

ter, brukte jeg observasjon som metode for datainnsamling. Observasjonene jeg gjorde er tredelt i form av video-, lyd- og skjermopptak. Bruk av opptak gjorde det også mulig å observere elevenes arbeid på papir og skjerm.

3.5.1 Observasjon

I kvalitative studier er observasjon en hyppig brukt metode for datainnsamling (Clark et al., 2021; Postholm & Jacobsen, 2018). Observasjon som forskningsmetode innebærer at «forskeren observerer deltakerne direkte i en naturlig setting» (Clark et al., 2021, s. 258, egen oversettelse). Observasjonene i denne studien kan sies å finne sted i naturlige settinger, fordi elevene kjenner såvel meg som hverandre, og de befinner seg i sine daglige læringsarealer.

Denne studien involverer et undervisningsopplegg som var nytt for elevene. I tillegg til programmering som et lite kjent konsept for elevene inneholder undervisningen en del selvstendig, men parvis, utforsking. Hadde dette vært en vanlig undervisning i helklasse hadde klassen naturligvis hatt tilgang til lærer og sannsynligvis også hatt behov for veiledning fra lærer. For å skape en mest mulig naturlig observasjonssetting, valgte jeg å ta rollen som lærer, i tillegg til observatør.

Skjermopptakene gjorde det mulig å følge med på elevenes programmeringsprosess blokk for blokk. Kombinasjonen av lyd- og skjermopptak gjorde at jeg kunne synkronisere elevenes programmering med deres respektive utsagn. Denne kombinasjonen gir et mer komplett bilde på elevenes iterative prosess enn om en av observasjonstypene ikke hadde blitt inkludert. Å undersøke elevers bruk av blokkprogrammering uten observasjoner fra skjermopptak ville potensielt ført til mangelfulle observasjoner og mindre pålitelige resultat. Videoopptakene lot meg observere elevarbeidet på papir. I begge øktene fikk elevene oppgaver på papir som de skulle skrive på. Samtykkeskjemaet for denne studien inkluderer ikke innsamlet elevarbeid, noe som krevde at jeg filmet det elevene gjorde og lagde egenskrevne kopier av elevenes arbeid. Videoopptakene lot meg også observere elevenes prosess med å plassere riktig blokkforklaring til riktig blokk i starten av den andre økta. Videoopptakene fungerte også som en støtte når lydopptaket ble svakt eller i situasjoner hvor det oppsto tvil om hvilken elev som snakket.

Gold (1958, henvist i Postholm & Jacobsen, 2018) skisserer fire forskjellige observatørroller for forskere, basert på grad av deltakelse og på grad av avstand til forskningskonteksten. Som forsker og lærerstudent i praksis, var graden av avstand alltid liten. Graden av deltakelse varierte imidlertid mellom liten og stor. Min rolle som observatør varierte dermed mellom deltaker-som-observatør og fullstendig deltaker. Valg av rolle var delvis planlagt og delvis en tilpasning til elevenes behov i undervisningsopplegget. Rollene jeg hadde var lik i både første og andre gjennomføring. I forkant av undervisningen var jeg bevisst på at min observatørrolle ville være ganske deltakende. Av den grunn tok jeg ikke i bruk et observasjonsskjema.

I første økt var jeg deltaker-som-observatør. Jeg veiledet elevene gjennom opplegget og stilte dem spørsmål underveis, for både å forstå og videreutvikle elevenes innspill underveis. Oppgavene i første økt var utformet slik at de la til rette for elevenes egne antakelser og utforsking, for å slik unngå for mye involvering fra min side. I den andre økta begynte

jeg begge gjennomføringene som deltaker-som-observatør, men inntok gradvis rollen som fullstendig deltaker, ettersom elevene hadde økende behov for faglig støtte. Den andre økta besto hovedsakelig av selvstendig utforsking, uten noe modellering fra min side. PRIMM-modellen vektlegger læreren som en stillasbygger og veileder i undervisningen (Sentance et al., 2019), noe som gjorde at jeg bevisst tillot meg større grad av involvering ettersom behovet for faglig støtte økte. Å unngå å gi elever faglig hjelp ved behov er også en forskningsetisk problemstilling: selv om jeg forsker på elevenes arbeid kan man diskutere hvorvidt det er etisk forsvarlig å la de utfordres uten tilstrekkelig støtte.

Postholm & Jacobsen (2018) poengterer at rollen som fullstendig deltaker kan skape utfordringer i observasjonen. Clark et al. (2021) trekker fram subjektivisme som et element i kvalitativ forskning som ofte er gjenstand for kritikk. Subjektivisme kan oppstå i flere av leddene i kvalitativ forskning, men vi kan særlig tenke oss at rollen som fullstendig deltaker i en observasjonssituasjon gjør at man som observatør går glipp av mange relevante observasjoner knyttet til problemstillingen. I denne studien er det likevel relevant å samtale med elevene om deres oppgaveløsing for å bedre forstå valgene de gjør underveis. En annen fare er at man bevisst eller ubevisst kun fanger opp momenter man selv mener er interessante. I slike tilfeller tar subjektiviteten for mye plass. For å øke mengden observasjoner og styrke observasjonenes gyldighet, tok jeg i bruk video-, lyd- og skjermopptak i hver undervisningsøkt. Jeg skrev noen stikkordsbaserte observasjonsnotater rett i etterkant av undervisningen, men hovedvekten av observasjonene er gjort ved å gjennomgå opptakene.

Clark et al. (2021) nevner noen fordeler ved lydopptak og tilhørende transkripsjon. Lydopptak minimerer tolkning av utsagn, det tillater grundig analyse av språk, det tillater gjentakende analyse av samme utsagn, i tillegg til at andre forskere får anledning til å vurdere de samme interaksjonen med egne øyne (Clark et al., 2021).

3.5.2 Innsamlet datamateriale

En fullstendig oversikt over innsamlet datamateriale kan sees i tabell 3.3. I tabellen refererer jeg til første gjennomføring som A og andre gjennomføring som B, hvor A1 og B1 er første økt i begge gjennomføringene. A2 og B2 er den andre økta i begge gjennomføringene. Første gjennomføring (A) besto av Brage/Nina og Maria/Fredrik. Andre gjennomføring (B) besto av Sander og Astrid/Hanne. Navnene er fiktive.

Tabell 3.3: Oversikt over innsamlet datamateriale

Innsamlet materiale	Mengde
Undervisningsøkter Avsatt 90 minutter	A1 og A2 (første gjennomføring) B1 og B2 (andre gjennomføring) Hver gjennomføring besto av to økter. **Totalt: 4**
Videopptak	45 minutter x 4 **Totalt: 180 minutter**
Lydopptak	50 minutter x 4 **Totalt: 200 minutter**
Skjermopptak	A1 Brage og Nina: -- A1 Maria og Fredrik: 53 minutter A2 Brage og Nina: 47 minutter A2 Maria og Fredrik: 48 minutter B1 Sander: 37 minutter B1 Astrid og Hanne: 38 minutter B2 Sander: 43 minutter B2 Astrid og Hanne: 43 minutter **Totalt: 309 minutter**
Transkripsjon (m/lyd + skjermbevegelser)	A1: 14 sider A2: 27 sider B1: 14 sider B2: 20 sider **Totalt: 75 sider**

Skjermopptak fra Brage og Nina i første økta (A1) viser ingen minutter i tabell 3.3. Dette skyldes at elevene ved et uhell stoppet opptaket i begynnelsen uten at jeg fikk det med meg. Denne delen av datamaterialet reduserte jeg riktignok vekk etter transkripsjonsfasen (se tabell 3.5 i 3.6.1 Transkribering).

Ser man nøye etter i tabell 3.3, legger man merke til at undervisningsøktene varte i 90 minutter. Fire undervisningsøkter skulle da ha tilsvart 360 minutter lyd- og videoopptak. I realiteten varte hver undervisningsøkt i omtrent 50 minutter. Dette var tilstrekkelig tid for å gjennomføre opplegget, samtidig som det ga rom for spontane praktiske utfordringer. Den resterende tiden deltok elevene i klassens opprinnelige undervisning.

3.6 Metode for analyse

Min studie svarer på *hvordan elever på 4. trinn bruker blokkprogrammering som semiotisk representasjonsregister i arbeid med tallfølger*. Clark et al. (2021) påpeker at induktiv tilnærming handler om å skape teori ut ifra observasjoner og funn i studien. Min problem-

stilling gir uttrykk for en induktiv tilnærming, fordi formuleringen tilsier at datamaterialet legger grunnlaget for svaret på spørsmålet.

Jeg begynte analyseprosessen med åpen koding, en induktiv tilnærming hvor man noterer ned alle observasjoner fra datamaterialet, uten å være styrt av teori (Nilssen, 2012). Deretter sorterte jeg kodene i kategorier og senere hovedkategorier som fanger essensen av materialet. Prosessen med åpen koding, som gjennom kategorisering gir hovedkategorier, knyttes til konstant komparativ analysemetode (se Nilssen, 2012). I kategoriseringsfasen tok jeg utgangspunkt i Duvals (2006) teori om semiotiske representasjoner for å skape kategorier som gjenspeiler både empiri og teori. Når en forsker tar hensyn til empiri og teori på samme tid har analyseprosessen en abduktiv tilnærming (Clark et al., 2021).

3.6.1 Transkribering

Transkribering er forskerens skriftlige gjengivelse av samtale (Nilssen, 2012). Transkribering er en viktig del av analyseprosessen, fordi forskeren blir kjent med materialet og får nye tanker om datamaterialet underveis i transkriberingen (Nilssen, 2012). For å ivareta elevenes personvern anonymiserte jeg elevene med fiktive navn før jeg begynte transkriberingen.

For å bli kjent med datamaterialet har jeg transkribert alt lydopptak, samt notert elevenes programmeringsprosess fra skjermopptakene. Jeg transkriberte totalt 200 minutter lydopptak og 309 minutter skjermopptak. Notatene fra skjermopptakene er skrevet inn sammen med transkripsjonen, på tidsmessig riktig steder. Nødvendige notater fra videoopptakene er også inkludert i transkripsjonene, slik som elevenes prosess med å plassere forklaringslapper på ulike blokker i den andre økta. I prosessen med å transkribere undervisningsøktene oppsto det spontane tanker og refleksjoner knyttet til noen av elevutsagnene. Disse notatene ble fortløpende nedskrevet som kommentar til de enkelte utsagnene. Nilssen (2012) trekker fram notering av tanker underveis som en av fordelene ved å selv transkribere eget materiale.

Transkripsjonskoder er symboler med tildelt mening, som brukes i transkribering for å påpeke blant annet nyanser i språket (Clark et al., 2021, s. 479). Transkripsjonskoder benyttes for å gjøre den ekte samtalen mest mulig realistisk i den nedskrevede versjonen. I forkant av transkriberingen tok jeg stilling til mengden materialet og problemstillingen i studien. Fokuset for studien er elevenes handlinger og samtale, ikke nyanser i elevenes språk og samtaler. Jeg valgte derfor å bruke få transkripsjonskoder. Se tabell 3.4 for en oversikt over transkripsjonskodene og deres betydning.

Tabell 3.4: Transkripsjonskoder

Kode	Forklaring
...	Pause på grunn av avbrytelse, men utsagnet fortsetter
understreking	Trykk på enkeltord
()	Observasjoner fra video eller skjermopptak noteres i parentes
..	(Pause på 1-2 sekund)

I tillegg til transkripsjonskodene, refererte jeg til den talende med de tre første bokstavene i navnet: Nin for Nina, Lær for lærer osv. Jeg benyttet også nummerering for hver linje

hvor en ny person snakket. Se helhetlig i figur 3.7. Nummerering i transkripsjonen gjorde det lettere å bli kjent med datamaterialet og å finne igjen enkeltdeler. Nummereringen gir også en indikasjon på hvor langt ut i undervisningen samtalen finner sted.

I undervisningsøktene foregikk noen samtaler i plenum, mens de fleste samtaler foregikk mellom elevene i læringspar. Elevene samtalte kontinuerlig. For å kunne analysere *hele* samtaler, ikke alle utsagnene blandet sammen, delte jeg transkriberingen i to deler: læringspar og plenumsamtale. Denne episodeinndeling gjorde det lettere å analysere sammenhengende samtaler. Inndelingen lot seg gjøre, fordi læringsparene arbeidet både selvstendig og adskilt fra hverandre.

For å oppnå større oversikt over programmeringsprosessene tok jeg skjermbilde av elevenes programforandringer og la dem samlet i et dokument, som en bildeserie. Hvert læringspar har en egen bildeserie. Bildeseriene har som hensikt å skape en oversikt over prosessen, men det fungerte også som en visuell støtte i lesingen av transkripsjonen, fordi bildene er nummerert og referert til i transkripsjonen. Transkripsjonene inneholder altså utsagn, hendelser på skjermen og bildenummer fra bildeserien. Se figur 3.7 for et utdrag fra transkripsjonen av samtalen mellom Brage, Nina og lærer (meg). BN33 i siste linje refererer til Brage og Ninas bilde nummer 33 i bildeserien. Det som er skrevet i parentes er hendelser på skjermen.

335	Lær	Hvis du har 1, for at det skal bli det neste tallet, må du plusse på 1.
336	Bra	Gange
337	Nin	Vi må ha pluss 2. Pluss 2..pluss..Vi må innpå pluss (går innpå matematikk). Pluss.
338	Bra	Men hvordan skal det plusse da?
339	Nin	1 pluss 1, 1 pluss 1. (Får oddetallsfølge på skjermen, BN33)

Figur 3.7: Eksempel fra transkripsjon

Ved å sette meg inn i datamaterialet på en systematisk og grundig måte i transkriberingsfasen, ble det tydelig for meg at mengden materiale kunne reduseres til to læringspar i stedet for fire, og en type økt i stedet for begge. Den første undervisningsøkta var mulig å redusere, fordi det i all hovedsak er den andre økta som relaterer til problemstillingen i min studie. Den første økta fungerte hovedsakelig som en ramme for å kunne gå videre til selvstendig programmering i læringspar i den andre økta. Hensikten med å redusere materialet var å skape en mer håndterlig mengde materiale uten at verdifull informasjon gikk tapt.

Tabell 3.5: Datamaterialet som ble analysert videre og redusert

Innsamlet materiale	Materialet som ble analysert videre	Redusert vekk
Undervisningsøkter Avsatt 90 minutter	A2 (første gjennomføring) B2 (andre gjennomføring) **Totalt: 2**	A1 (første gjennomføring) B1 (andre gjennomføring) **Totalt: 2**
Videopptak	A2: 45 minutter B2: 45 minutter **Totalt: 90 minutter**	A1: 45 minutter B1: 45 minutter **Totalt: 90 minutter**
Lydopptak	A2: 50 minutter B2: 50 minutter **Totalt: 100 minutter**	A1: 50 minutter B1: 50 minutter **Totalt: 100 minutter**
Skjermopptak	A2 Brage og Nina: 47 minutter B2 Astrid og Hanne: 43 minutter **Totalt: 90 minutter**	A1 Brage og Nina A1 Maria og Fredrik A2 Maria og Fredrik B1 Sander B1 Astrid og Hanne B2 Sander **Totalt: 219 minutter**
Transkripsjon (m/lyd + skjermbevegelser)	A2 (Brage og Nina + felles): 17 sider B2 (Astrid og Hanne + felles): 14 sider **Totalt: 31 sider**	A1 og B1 (28 sider) A2 Maria og Fredrik (10 sider) B2 Sander (6 sider) **Totalt: 44 sider**

Jeg reduserte vekk observasjoner fra Sander og Maria og Fredrik, såvel som førsteøktene for alle par (A1 og B1). Jeg sto dermed igjen med den andre økta for Brage og Nina og den andre økta for Astrid og Hanne. Jeg valgte å fokusere analysen på Brage og Nina og Astrid og Hanne, fordi disse elevparene brukte blokkene på samme måte som de andre elevene, samtidig som de kom med flere innspill knyttet til oppgaven og programmeringen. Siden jeg gjennomførte datainnsamling i to runder, med endringer fra første til andre gjennomføring, ønsket jeg et læringspar fra hver gjennomføring. I tillegg utfyller de to læringsparene hverandre. Astrid og Hanne gjør flere presise matematiske betraktninger med hensyn til programmet, mens Brage og Nina prøver ut flere løsninger og kommer sånn sett nærmere riktig løsning.

Det kan kanskje virke som om i overkant store deler av materialet er redusert, men de 31 sidene med transkripsjon som jeg analyserte videre, er konsentrert materiale. Elevene arbeidet med oppgaven stort sett sammenhengende hele økta og samtalene om oppgaven pågikk kontinuerlig.

3.6.2 Konstant komparativ metode

For å analysere alt datamaterialet tok jeg i bruk konstant komparativ metode. Konstant komparativ metode er opprinnelig en analysemetode av Strauss & Corbin (1990 i Clark et al., 2021) som ofte relateres til forskningsdesignet «grounded theory» (Nilssen, 2012). Metoden har som hensikt å la datamaterialet være grunnlaget for nye teoretiske betraktninger (Nilssen, 2012).

Duvals (2006) opprinnelige rammeverk om semiotiske representasjonsregistre betrakter ikke programmering. Analyse av materialet i min studie vil kunne bidra til informasjon om hvordan vi kan forstå blokkprogrammering som representasjonsregister i matematikk. Forskning som betrakter blokkprogrammering som register, er svært begrenset. Det finnes derfor ikke etablerte koder eller kategorier jeg kan lete etter for å undersøke elevers bruk av registeret. En induktiv tilnærming blir av den grunn mest hensiktsmessig for å undersøke elevers bruk av blokkprogrammering som register. Jeg valgte konstant komparativ analysemetode som den overordnete induktiv tilnærmingen til datamaterialet, fordi jeg ønsker å tilføre nye betraktninger til Duvals registerteori.

Analyse med konstant komparativ metode består av de tre stegene åpen koding, aksial koding og selektiv koding (Strauss & Corbin, 1990 i Clark et al., 2021). Stegene bør foregå som en iterativ prosess for å best fange essensen av materialet som helhet (Clark et al., 2021). Konstant komparativ metode refererer til en kontinuerlig sammenligning av koder og kategorier i materialet for å finne ut hvilke kategorier eller indikatorer som best belyser materiale (Clark et al., 2021). Åpen koding er det første steget. I åpen koding skal forskeren sette navn på hva som skjer i datamaterialet uten å la teoretiske betraktninger styre hva man leter etter (Nilssen, 2012). Problemstillingen blir ledende for hva man ser etter, men i motsetning til deduktiv tilnærming, skal man ikke lete i materialet etter koder som allerede eksiterer fra etablerte teorier (Nilssen, 2012). Etter den første kodingsprosessen sitter man igjen med mange koder som må sorteres videre i kategorier for å gjøre materialet håndterlig (Nilssen, 2012). Jeg analyserte elevparene hver for seg og satt igjen med 112 koder for Brage og Nina og 82 koder for Astrid og Hanne. Kodene var både basert på elevenes utsagn og programmeringsprosessen fra skjermopptakene. Jeg valgte å gjennomføre kodingen som en tredelt prosess hvor jeg undersøkte transkripsjonene og bildeseriene hver for seg, og til slutt i samspill. Jeg valgte å sammenligne utsagn med handling for å kunne påpeke om elevene gjør skifter mellom registre.

Ved å studere programmeringsprosessen isolert sett oppdaget jeg trender knyttet til valg og bruk av blokker i elevenes programmer. For å enklere studere elevenes programmeringsprosess på skjermen lagde jeg som nevnt en bildeserie for hvert læringspar som inneholdt alle programvariantene de skapte. Bildeseriene gjorde det mulig å studere elevenes bruk av blokker på en oversiktlig måte, samt at bildeseriene fungerte som en visuell støtte i gjennomlesing av transkripsjonene. Nilssen (2012) skriver at analyseredskaper, som en bildeserie, er til hjelp i kodeprosessen.

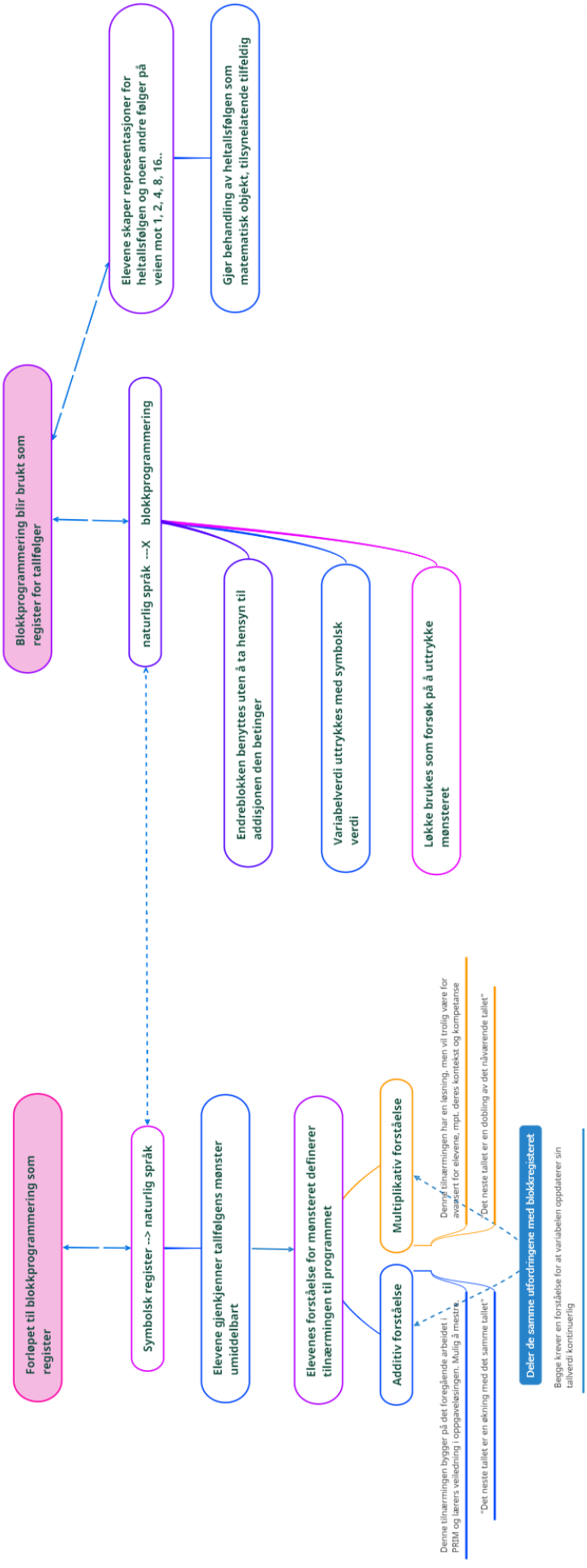
Etter fasen med åpen koding ønsket jeg å knytte datamaterialet til Duvals (2006) rammeverk om semiotiske representasjoner. For å oppnå en oversikt over hvorvidt elevene faktisk skapte semiotiske representasjoner i programmeringsmiljøet, utformet jeg fire kategorier som er direkte knyttet til Duvals (2006) teori:

- Programmet er ikke en semiotisk representasjon for matematikk
- Programmet er ikke en semiotisk representasjon for tallfølger
- Programmet er en semiotiske representasjon for tallfølger
- Programmet er en semiotisk representasjon for den konkrete tallfølgen

Utformingen av de fire kategoriene eksemplifiserer at analysen pendler mellom empiri og teoretisk rammeverk, som er et av kravene til designforskning (Prediger et al., 2015). En slik iterativ prosess sammenfaller med abduktiv forskningstilnærming. Analysen viste tydelig at elevene brukte blokkprogrammering som semiotisk register hele tiden, med noen få unntak. Unntakene relaterte til bruk av musikkblokk, som spiller av lyd, og mangel på synlighet. Et program kan utføre en handling uten at noe vises på micro:bit'ns skjerm, men ved bruk av «vis tall ()» eller «vis tekst ()» kan man gjøre programmets handlinger synlig. Jeg anser synlighet i programmet som en naturlig del av 4. trinns elevers bruk av blokkprogram som representasjon for matematikk. Selv om et program kan utføre matematiske operasjoner uten å vise noe på skjermen, er det ikke sikkert en 4. trinns elev vil forstå det. Programmering er relativt nytt for elevene i denne studien, særlig nytt er programmering som matematisk representasjon. Etter min mening er det hensiktsmessig å inkludere synlighet i blokkprogram som en betingelse for å kalle blokkprogrammet for en matematisk representasjon.

Prosessen med å kategorisere alle kodene kalles aksial koding (Nilssen, 2012). I dette steget skal forskeren se kodene i sammenheng med hverandre og navngi kategorier som omtaler hovedaspektene ved kodene (Nilssen, 2012). I denne studien sammenligner jeg kodene og de tentative kategoriene i flere omganger. For hver gang jeg utformet en kategori studerte jeg kodene på nytt for å vurdere om kategorien faktisk gjenspeilet kodene eller ikke. De til sammen 194 kodene oppsummerte jeg i et digitalt tankekart, som jeg videre komprimerte til et mindre og mer konsist tankekart. Tankekartene kan i likhet med bildeseriene anses som analyseredskap.

Figur 3.8 viser en oversikt over datamaterialet i form av et digitalt tankekart. Dette tankekartet var forløperen til hovedkategoriene som oppsto i den siste fasen, selektiv koding. Jeg ønsker å inkludere denne figuren for å gi et innblikk i deler av analyseprosessen. Tankekartet viser et øyeblikksbilde av den aksiale kodefasen.



Figur 3.8: I prosessen med aksial koding

Overordnet organiserte jeg tankekartet i to deler: før programmeringen begynte og i programmeringsprosessen. Disse to delene er de rosa boblene øverst i figur 3.8. Helt til venstre i tankekartet oppsummeres elevenes arbeid med den symbolske tallfølgen, altså før blokkprogrammering ble tatt i bruk. Der kan vi blant annet se at tallfølgens struktur forstås på to ulike måter av elevene: additivt og multiplikativt. Den blå gjennomfargede boksen langt nede, under elevenes mønsterforståelse, påpeker hvordan elevenes tolkning av mønsteret ikke påvirker deres bruk av registeret. Med andre ord benytter elevene registeret på samme måte, uansett hvilken matematisk forståelse de la til grunn for den symbolske tallfølgen.

I resten av tankekartet, delen hvor blokkprogrammering brukes som register, kategoriserer jeg enkelte gjennomgående trekk ved hvordan elevene tar i bruk blokkprogrammering som register. Denne delen av tankekartet viser elevenes misforståelser omkring blokkprogrammering.

Den siste fasen i konstant komparativ metode er den selektive kodingen (Nilssen, 2012). I denne fasen skal kategoriene sammenlignes og relateres til hverandre, hvor målet er å finne en kjernekategori som oppsummerer alle de foregående kategoriene (Clark et al., 2021). I analyse av datamaterialet oppsto et behov for to hovedkategorier, i stedet for én kjernekategori.

Fra tankekartet i figur 3.8 kan vi se nærmere på «naturlig språk —X blokkprogrammering» i midten av tankekartet. Denne boblen refererer til hvordan elevene uttrykte muntlig hvordan de ville at tallene skulle øke, uten at de lyktes fullstendig med omdanningen til blokkprogrammering som register. Boblen har tre underkategorier:

- Variabelverdi uttrykkes som symbolsk tallverdi
- Endreblokk benyttes uten å ta hensyn til addisjonen den betinger
- Løkke benyttes som forsøk på å uttrykke mønsteret i tallfølgen

Disse tre måtene å bruke blokkprogrammering på er alle eksempler på gjennomgående trekk i datamaterialet. Begge læringsparene brukte blokkprogrammering på denne måten i arbeid med å representere den symbolske følgen {1, 2, 4, 8, 16, ...}. Underkategoriene er gjennomgående i datamaterialet, fordi de gjaldt for alle elevene og elevene brukte blokkprogrammering på disse måtene uten unntak. I prosessen med selektiv koding utviklet jeg hovedkategorien «Betingelser i blokkprogrammering som register». Denne hovedkategorien tar hensyn til alle de tre kategoriene samtidig.

Det finnes også andre betingelser i blokkprogrammering som register. I den åpne kodingen av datamaterialet fant jeg ytterligere to koder som omtalte betingelser i registeret som elevene forholdt seg til. Kronologi og synlighet i programmet. Kronologi i programmet refererer til at en datamaskin leser et program eller instruksjonene fra topp til bunn, uten å legge til nyanser underveis. Rekkefølgen elevene plasserer blokkene er derfor avgjørende for programmets resultat. Synlighet i programmet er en betingelse som jeg har definert selv.

Kronologi og synlighet var aspekter som elevene mestret i prosessen med å bruke blokkprogrammering som register. De få gangene elevene ikke brukte riktig kronologi eller synlighet, ble dette korrigert av elevene underveis. Synlighet og kronologi er også to betingelser som

elevene har forholdt seg til alle øktene de har programmert med MakeCode. Som nevnt i 3.2 Beskrivelse av utvalg og kontekst, hadde elevene gjennomført flere programmeringsøkter før de gikk i gang med undervisningen for dette prosjektet. Det er derfor rimelig å anta at kronologi og synlighet er noe de hadde kjennskap til og mestret til en viss grad. Bruk av løkke og variabler var imidlertid mer ukjent og noe de manglet erfaring med.

For studiens transparens er det viktig å nevne de betingelsene ved blokkprogrammering som elevene mestret, men jeg vil legge mer vekt på betingelsene som elevene strevde med, da det var disse aspektene som påvirket elevenes arbeid mest. Et aspekt ved å bruke et representasjonsregister er å gjøre omdanning til og fra andre registre (Duval, 2006, 2017). Elevene i denne studien kom ikke helt i mål med omdanning fra symbolsk register eller naturlig språk til blokkprogrammering. Analysen av datamaterialet, ved bruk av konstant komparativ metode, indikerer at elevene ble hindret i å lykkes med omdanning til blokkprogrammering som register, på grunn av enkelte betingelser som ikke ble oppfylt.

«Betingelser i blokkprogrammering som register» er en hovedkategori som har et nærgående blikk på datamaterialet. I prosessen med selektiv koding fikk jeg også behov for å omtale omdanning og hvordan det skjer omdanning mellom noen registre, og ikke andre. Dette aspektet har et mer overordnet blikk på datamaterialet, slik det også framstilles i tankekartet fra figur 3.8. Enda en hovedkategori ble derfor utformet, «omdanning mellom ulike registre». Etter den selektive kodingen sto jeg igjen med to hovedkategorier med henholdsvis 2 og 3 underkategorier:

- Omdanning mellom ulike registre
  - Fra symbolsk register til naturlig språk
  - Fra blokkprogrammering til naturlig språk
- Betingelser i blokkprogrammering som register
  - Elevene uttrykker variabelverdi som symbolsk tallverdi
  - Elevene benytter endreblokk uten å ta hensyn til addisjonen den betinger
  - Elevene bruker løkke for å uttrykke tallfølgens økning

3.7 Forskningens troverdighet

I kvalitativ forskning er det hensiktsmessig å snakke om forskningens kvalitet ved å diskutere forskningens troverdighet (Nilssen, 2012). Troverdig forskning skal overbevise leseren om at studien gir en riktig framstilling av forskningens kontekst (Nilssen, 2012). Konteksten omfatter elevenes bakgrunn, undervisningsopplegget, situasjonen knyttet til datainnsamling, min egen kjennskap til elevene, med mer. Forskning er både prosessen og resultatene (Postholm & Jacobsen, 2018, s. 219).

For å styrke studiens troverdighet kan man gjøre tiltak tilknyttet studiens transparens (Nilssen, 2012). Transparens handler å synliggjøre forskningsprosessens helhet (Nilssen, 2012, s. 42). Med andre ord, gi leseren mulighet til å ta del i forskningen gjennom å forstå hvilke valg som er gjort og hvilke tolkninger som ligger til grunn for valgene. Denne studien etterstreber transparens ved å blant annet vise leseren et øyeblikksbilde fra kategoriprosessen i analysen, se figur 3.8. Tankekartet gir leseren mulighet til å både forstå deler

av min tolkningsprosess med å finne hovedkategorier, og samtidig får leseren anledning til å være medtolker. I presentasjon av funn (4 Resultat) gir jeg leseren anledning til å tolke eksemplene fra datamaterialet opp mot mine hovedkategorier og underkategorier. Eksemplene gir leseren konkrete innblikk i datamaterialet, samtidig som leseren selv kan ta stilling til hvorvidt eksemplene har relevans til den gitte kategorien.

Forskeren må ha et bevisst forhold til sin subjektivitet, slik at denne blir inkludert i studien med hensikt å vise leseren hvilken kontekst funnene kan bli forstått i (Merriam, 2002 i Postholm & Jacobsen, 2018, s. 220). Min subjektivitet er en faktor i flere deler av studien, men særlig med tanke på undervisningsopplegg og datainnsamling. Ved å være bevisst min egen subjektivitet vil jeg oppnå ytterligere transparens i studien, fordi det i større grad blir mulig å forstå mitt tolkningsgrunnlag. Under datainnsamlingen hadde jeg rollen som observatør og lærer samtidig. Spørsmålene jeg stilte, og min generelle tilnærming til elevene, vil påvirke deres arbeid. En annen lærer ville potensielt gjennomført opplegget med en annen tilnærming, men så lenge jeg synliggjør min hensikt og min rolle har leseren av studien anledning til å forstå forskningens kontekst. Å gi en detaljert beskrivelse av forskningens kontekst er også en vesentlig del av designforskning (Gravemeijer & Cobb, 2006; Prediger et al., 2015), som er forskningsdesignet i denne studien.

Som nevnt i 3.2 Beskrivelse av utvalg og kontekst, kjenner jeg elevene som deltar i denne studien, i tillegg til at jeg både er den som har designet undervisningsopplegget og den som har rollen som lærer i datainnsamlingen. Det er ikke uvanlig at forskere i kvalitative studier forsker i sin egen kontekst (Nilssen, 2012, s. 138).

Min subjektivitet i møte med elevene kan påvirke studiens troverdighet både positivt og negativt. Rollen som deltaker-som-observatør og fullstendig deltaker ga meg fordelen av å få nærgående kjennskap til elevenes arbeid og refleksjoner. Bruk av video-, lyd- og skjerm-opptak fungerte også som et tiltak for å styrke studiens troverdighet ved å minimere egne subjektive tolkninger av selve gjennomføringen. Ved å se gjennom alle opptakene, særlig i forbindelse med transkriberingen, fikk jeg flere muligheter til å revurdere egne tolkninger. I analyseprosessen fungerte også konstant komparativ metode som en ramme for å belyse essensen i datamaterialet, uten at jeg som forskersubjekt selv valgte ut enkeltelementer jeg mente var interessante. Funnene fra analysen er interessante elementer fra materialet, men de har oppstått gjennom en etablert analysemetode som har veiledet min prosess i utviklingen av kategorier og hovedkategorier.

Min aktive rolle i utforming og gjennomføring av undervisningsopplegget har også sine negative sider for studiens troverdighet. En rekke valg ble tatt i prosessen med å utforme opplegget, særlig knyttet til valg av blokkprogram og tilhørende tallfølge elevene skulle skape i andre økt av opplegget. Elevene kom ikke i mål med oppgaven og flere uttrykte frustrasjon overfor oppgaven underveis. Siden oppgaven sånn sett ikke var optimalt tilpasset elevenes programmeringsferdigheter, kan man stille seg kritisk til valg av oppgave. Oppgaven ble definert av meg, uten diskusjon med andre lærere eller studenter. Den delen av datamaterialet som analyseres i min studie baserer seg også utelukkende på denne oppgaven. Flere antall oppgaver, med potensielt større tilpasningsmuligheter med hensyn til ferdigheter, kunne økt studiens troverdighet.

En designstudie skal som nevnt ikke generaliseres til å gjelde alle andre lignende undervisningskontekster, men resultatene av forskningen skal danne grunnlag for å anvende

funnene til andre situasjoner (Gravemeijer & Cobb, 2006). Én oppgave og tallfølge alene kan potensielt oppfattes som for lavt antall til å kunne relateres til andre situasjoner som bruker blokkprogrammering som register for tallfølger.

PRIMM-modellen baserer seg på at programmet i det siste steget skal ha likheter med programmet i de tidligere stegene, men likevel være et annet program (Sentance et al., 2019). Å utforme et program og tilhørende tallfølge som ikke overskred elevenes ferdighetsnivå i programmering, var imidlertid vanskelig. Ved å være åpen om utfordringer knyttet til utforming av oppgave, styrkes studiens transparens.

3.8 Etikk og personvern

Nilssen (2012) trekker frem blant annet informert samtykke og anonymisering som etiske betraktninger i forskning. Når man ønsker å behandle personvernopplysninger i forskning, eksempelvis navn, stemme eller bilde, er man nødt til å melde inn prosjektet til «Sikt[2]» (Postholm & Jacobsen, 2018). Sikt behandler innmeldingen og vurderer hvorvidt datainnsamling og behandlingen av materialet er i tråd med personvernhensyn (Sikt, 2022). I min studie samler jeg inn datamateriale i form av video-, lyd- og skjermopptak. Skjermopptak er ikke spesifisert i innmeldingen til Sikt, men skjermopptaket fanger ingen personvernopplysninger på skjermen. Noen skjermopptak foregikk med lyd, men lydopptak er meldt inn til Sikt. Elevene som deltar i denne studien, og deres foresatte, har alle samtykket til at jeg behandler disse personvernopplysningen om elevene.

Sikt har vurdert at studiens innsamling og behandling av personvernopplysninger foregår på lovlig vis (se vedlegg 7). Min studie imøtekommer kravene til personvern ved informert samtykke og anonymisering av alt datamaterialet. Video- og lydopptak slettes ved fullstendig anonymisert transkripsjon og observasjonsnotater. Elevarbeid fra undervisningsopplegget har jeg gjengitt med min egen håndskrift og uten elevenes navn. Materialet i sin helhet vil slettes når prosjektet er over juni 2023.

I prosessen med å anonymisere datamaterialet lagres rådataene i en personlig skylagring gjennom NTNU. Skjermopptakene fra elevenes arbeid på Chromebook overførte jeg til minnepenn, og deretter direkte inn på skylagringen. Innholdet på minnepennen slettet jeg rett etter overføring til skylagring. Datamaterialet fra kamera og lydopptaker overførte jeg direkte fra innsamlingsverktøyet og inn i skylagringen.

Personer som deltar i forskning skal ha anledning til å trekke seg når som helst (Nilssen 2012). Dette er spesifisert i samtykkeskjema som vi i studentgruppa sendt ut til alle foreldre på trinnet hvor våre datainnsamlinger foregikk. Ved forskning på barn er også barnets eget samtykke nødvendig (Nilssen, 2012). I forbindelse med studien spurte jeg flere elever om å delta, også flere enn de som faktisk deltok. De elevene jeg spurte, var elever med foreldre som samtykket på vegne av barnet. Når jeg spurte elevene om de ønsket å delta presiserte jeg to ting: det er lov å si nei og aktiviteten vi skulle gjøre var nært knyttet til matematikkundervisningen de uansett skulle ha. Jeg spurte alltid elevene direkte, uten andre elever umiddelbart til stede. Nilssen (2012) poengterer at forskningsdeltakere må ha en reell mulighet til å si nei. Selv om jeg presiserte at de kunne si nei, kan elevene oppleve et press enten fra meg eller ved at andre elever deltar. Kanskje deltar for eksempel elevens

[2]Tidligere NSD

beste venninne i studien. Ved å spørre elevene uten andre elever umiddelbart til stede, kan det tenkes at elevene i større grad fikk en reell mulighet til å si nei, enn om jeg hadde spurte klassen i plenum.

Som nevnt presiserte jeg overfor elevene at undervisningen vi skulle gjøre lignet på undervisningen de uansett ville hatt i de gitte timene. Ved å relatere undervisningsopplegget til klassens matematikkundervisning, og samtidig gjennomføre datainnsamling i klassens matematikkøkter, forsøkte jeg å minimere elevenes belastning av å delta i studien. Unngå skade og merbelastning er et av de etiske betraktningene Nilssen (2012) løfter frem. En av elevene i denne studien var bekymret for at den andre økta i undervisningsopplegget ville krasje med svømmeundervisningen klassen skulle ha. Eleven hadde misforstått ukeplanen, men eksemplet hans trekker fram en potensiell skade eller merbelastning for eleven om denne opplysningen hadde vært riktig.

Postholm & Jacobsen (2018) påpeker at forskere forplikter å gi en riktig gjengivelse av datamaterialet, slik det faktisk foregikk. Forskningsdeltakere sine uttalelser og handlinger skal gjengis slik de faktisk ble sagt og gjort. I denne studien har jeg blant annet vært opptatt av å være pålitelig i transkriberingen. Jeg har sikret at hver uttalelse noteres med nøyaktig formulering og tvil knyttet til lav lyd eller eksternt støy har jeg presisert eksplisitt i transkripsjonen. Pålitelige transkripsjoner styrker også studiens troverdighet.

Denne studien er en del av forskningsprosjektet LAB-TEd og i den forbindelse er vi fire studenter som har samarbeidet om innsamling av datamaterialet. Siden vi samlet datamaterialet på den samme skolen og det samme trinnet meldte vi inn våre studier som en samlet innmelding til Sikt. Ser man på innmeldingen til Sikt i vedlegg 7 vil man se at det er en av de andre studentene som har sendt inn studiene til Sikt, ikke meg. Riktignok kan man i samtykkeskjemaet utsendt til foreldrene i vedlegg 6 se at alle studentene i gruppa er listet opp nederst, inkludert meg selv.

I de nasjonale forskningsetiske retningslinjene for samfunnsvitenskapelig forskning, står det at resultater fra forskning skal formidles med resten av samfunnet (NESH, 2021). Min studie blir publisert offentlig, men studien blir også delt med alle involverte parter i LAB-TEd-prosjektet, som er det overordnede forskningsprosjektet jeg og min studie er en del av.

4 Resultat

Denne studien søker å besvare problemstillingen *hvordan bruker elever på 4. trinn blokkprogrammering som semiotisk representasjonsregister i arbeid med tallfølger?* I dette kapittelet presenterer jeg funnene fra analysen, med hovedkategoriene *omdanning* og *betingelser* som utgangspunkt. For hver av kategoriene følger underkategorier som vil bli belyst etter tur, med eksempler fra datamaterialet.

4.1 Omdanning mellom ulike registre

Duval (2004) omtaler omdanning som en veksling mellom semiotiske representasjoner som krever et registerskifte. Utgangspunktet for undervisningsopplegget var at elevene skulle gjøre omdanning fra en symbolsk representert tallfølge til den samme tallfølgen representert i et blokkprogram. Elevene i min studie gjør omdanning fra symbolsk register til naturlig språk, og fra blokkprogrammering til naturlig språk. Elevene mestret ikke en fullstendig omdanning fra det symbolske registeret eller naturlig språk, til blokkprogrammering som register.

4.1.1 Omdanning fra symbolsk register til naturlig språk

Elevene gjør omdanning fra symbolsk register til naturlig språk i begynnelsen av den andre økta. Før programmeringen begynte fikk elevene vite tallfølgen gjennom den symbolske representasjonen {1, 2, 4, 8, 16, ...}. Begge elevparene uttrykte muntlig hvordan tallfølgen ville fortsette, eller var bygd opp, etter å ha sett på tallfølgen som symbolsk representasjon. Når elever omtaler et matematisk objekt ved bruk av ord benytter de representasjonsregisteret naturlig språk (Duval, 2006). Ved å studere en symbolsk representert tallfølge, som er det matematiske objektet i dette tilfelle, og deretter sette ord på egenskaper ved tallfølgen, gjør de omdanning fra symbolsk register til naturlig språk som register.

97	Lær:	(...) Det finnes en tallfølge og den er slik: 1, 2, 4, 8, 16, ... Jeg gjentar 1, 2, 4, 8, 16, ...
98	Bra:	32!
		[...]
102	Bra:	Den må dobles med alt

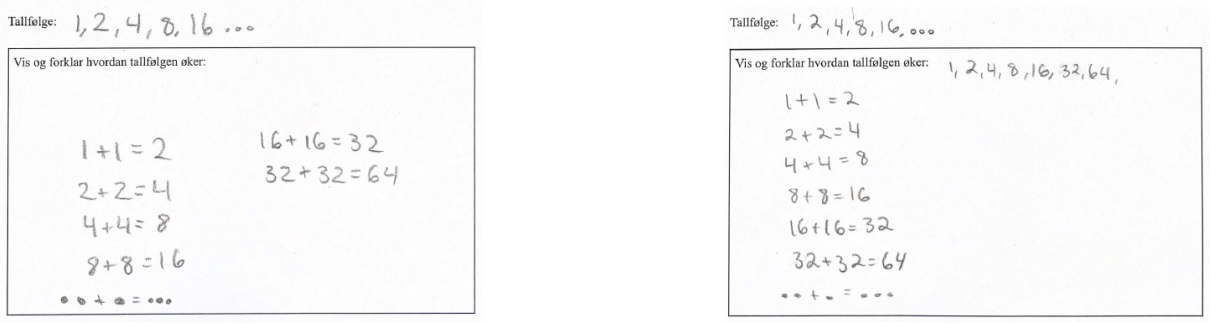
Samtaleutdraget viser at Brage umiddelbart gjenkjenner det neste tallet i følgen, samtidig som han like etterpå gir en verbal forklaring på mønsteret i tallfølgen, «den må dobles med alt».

Det neste samtaleutdraget viser Brage og Ninas innledende samtale når de begynte programmeringen. Denne fasen foregår i registeret naturlig språk samtidig som det de snakker om er den symbolske tallfølgen og blokkprogrammering. I dette tilfelle er læringsparet i tre registre samtidig.

110	Nin:	Hvor mye dobler man da?
111	Bra:	Man dobler med..man dobler bare. Man ganger med 2
112	Nin:	Da må vi jo ha noe med gange
113	Bra:	Og vent! Matematikk (i MakeCode)..Ehm..hvor er gange?
114	Nin:	2..1 gange 2 (operasjonsblokk med 1x2 settes inn løkkeverdien)
115	Bra:	Ehm..nei..det er noe med matematikk. (Skjermen viste ingenting pga. manglende innhold i løkka, kaster vekk operasjonsblokken). Vet ikke helt hva
116	Nin:	Men det er vanskelig å vite hva (til lærer). Vi vet hva det blir, men vi vet ikke hvordan vi finner ut av det (i programmeringsmiljøet)
117	Lær:	Skjønner..men det er bare fint at vi tenker litt

Brage og Nina fra første gjennomføring gikk rett i gang med programmeringen etter noen raske betraktninger om tallfølgens mønster. Astrid og Hanne fra andre gjennomføring fikk anledning til å beskrive tallfølgen på papir før de gikk i gang med programmeringen. Se figur 4.1 for Astrid og Hannes beskrivelse av mønsteret, gjengitt av forsker. Jentene gjorde seg også noen betraktninger av tallfølgens mønster, før de beskrev mønsteret på papiret. I dette samtaleutdraget er også Sander med:

115	San:	Det starter med 1. Det dobbelte av 1 er 2
116	Han:	Og ja! Dobbelte av 2 er 4. Dobbelte av 4 er 8, og det dobbelte av 8 er 16!
117	Lær:	Mhm
118	Han:	Jeg er så smart
119	Ast:	Og det dobbelte av 16 er 32!
120	Lær:	Ja! Kan du prøve å formulere det på en måte? Eller vise det med en tegning eller et eller annet i den boksen?
121	San:	Det..det..
122	Ast:	32+32 er 64. 64+64 vet jeg ikke



(a) Astrids beskrivelse av følgen (b) Hannes beskrivelse av følgen

Figur 4.1: Astrid og Hannes beskrivelse av tallfølgen {1, 2, 4, 8, 16, ...}

Samtaleutdraget med jentene og Sander viser hvordan de muntlig gir en beskrivelse av mønsteret i tallfølgen etter å ha sett den symbolske representasjonen av følgen. En muntlig beskrivelse av tallfølgen er en representasjon for tallfølgen i registeret naturlig språk. Elevene anser mønsteret som dobling, men i beskrivelsen på papiret uttrykker de mønsteret som gjentatt addisjon. Jentenes skriftlige, og symbolske, beskrivelse av mønsteret i tallfølgen baserer seg på at det neste tallet oppnås ved å addere det tallet man har nå, med seg selv. For å oppnå 2 må man ta 1+1. For å oppnå 4 må man ta 2+2 osv.

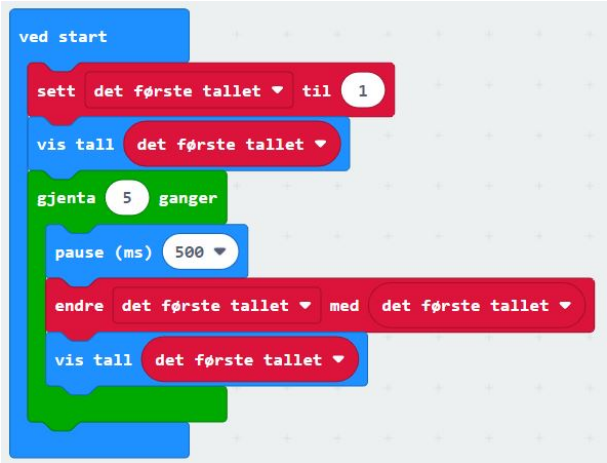
Jentenes beskrivelser på papir kan anses som en behandling av den symbolske tallfølgen, fordi de representerer tallfølgen på en annen måte, med + og =, men representasjonen er fortsatt i det symbolske registeret.

4.1.2 Omdanning fra blokkprogrammering til naturlig språk

Elevene i denne studien oppnår ikke fullstendig omdanning fra symbolsk register eller naturlig språk til blokkprogrammering som register, hvilket var hovedformålet med prosjektet, men noen av elevene gjør omdanning motsatt vei, fra blokkprogrammering til naturlig språk.

Etter omtrent 30 minutter med sammenhengende prøving og feiling presenterte jeg riktig løsning overfor elevene, uten å avsløre at løsningen var riktig. I gjennomføringen med Astrid og Hanne fjernet jeg på forhånd simulatoren i MakeCode, slik at elevene ikke kunne se resultatet av programmet, kun selve programmet. Elevene fikk anledning til å kommentere om programmet fungerte og i så fall hvorfor. Når elever ser et program og tolker det slik at de kan sette ord på hva det gjør, gjør de omdanning fra blokkprogrammering som register til naturlig språk som register.

Et eksempel på omdanning fra blokkprogrammering til naturlig språk kan vi se i dette samtaleutdraget mellom lærer og Hanne:



Figur 4.2: Fasit

479	Lær:	Hva er det første tallet..før vi har trykket på en måte?
480	Han:	1
481	Lær:	Endre 1 med 1? (peker på endreblokka)
482	Han:	Jaa
483	Lær:	Hva betyr det?
484	Han:	2
485	Lær:	2 ja. Okei, så da vises det 2 når jeg har trykket en gang
486	Han:	Men da må du endre 2 med det første tallet som blir et annet tall. Da blir..da er det første tallet blitt 2. 2+2 og da blir det åt..fire. Nei, jo 4. Så da blir det 4+4 og så bla bla bla bla bla

Astrid og Hanne fikk lov til å bruke knapp som inndata i programmet, noe Brage og Nina ikke behøvde. Begge parene hadde utfordringer med å skape et program som represen-

terte tallfølgen, men Brage og Nina mestret etter hvert å bruke alle nødvendige blokker, bortsett fra å sette inn variabelblokka «det første tallet» på verdiplassen til endreblokka. Astrid og Hanne hadde lite progresjon i startfasen, derfor bestemte jeg at de kunne kopiere programmet fra den første økta og heller modifisere dette programmet for å oppnå en representasjon for den nye tallfølgen.

Samtaleutdraget over viser at Hanne gjør omdanning fra blokkprogrammering til naturlig språk, ved å sette ord på hva slags matematikk programmet uttrykker. Omdanning den andre veien, fra naturlig språk eller symbolsk register til blokkprogrammering, var det ingen av elevene som helt og fullt oppnådde. Som jeg demonstrerte i 4.1.1, klarte elevene raskt å sette ord på tallfølgens mønster. Med dette som utgangspunktet, og den første økta med PRIM som erfaring, var tanken at elevene skulle klare å skape en semiotisk representasjon for tallfølgen, som et blokkprogram. Dette, videre omdanning fra naturlig språk til blokkprogrammering, skjedde imidlertid ikke.

4.2 Betingelser i blokkprogrammering som register

Innunder *Betingelser i blokkprogrammering som register* tar jeg for meg tre utfordringer som sto i veien for elevenes omdanning til blokkprogrammering som register. Utfordringene hadde å gjøre med enkeltblokker, og betingelsene knyttet til hvordan de skal brukes i blokkprogrammering.

4.2.1 Elevene uttrykker variabelverdi som symbolsk tallverdi

Begge elevparene uttrykker konsekvent variabelverdier som symbolske tallverdier. En variabel er en navngitt verdi som lagres som et minne i programmet (Usiskin, 1988). I et blokkprogram som representerer tallfølger, er det nødvendig å benytte variabler. I figur 4.2 kan vi se at variabelblokka «det første tallet» er satt inn i verdiplassen til endreblokka. Variabelen «det første tallet» endrer verdi underveis, og når man skal uttrykke den nåværende verdien til variabelen, i programmet, må man sette inn variabelblokka. Elevene i studien setter inn den faktiske tallverdien til variabelen i programmet, i stedet for å sette inn variabelblokka. Hvis variabelen per nå har verdien 1 og man ønsker å bruke den verdien må man sette inn variabelblokka, ikke det symbolske tallet 1. Tallsymbolet 1 er uavhengig av variabelverdien selv om begge per nå har verdien 1. Elevene viser både i uttalelsene og i programmeringen at de ønsker å uttrykke variabelverdien, men det er konsekvent symbolske tallverdier som benyttes. Nedenfor presenterer jeg to eksempler på bruk av symbolsk tallverdi i programmet, når uttalelsen tilsier bruk av variabelverdi.

168 Lær: Brage, hvis du tenker kun på den tallfølgen som er der (på papiret). Hva er det som ganges med 2?
169 Bra: Ehm..det først..det første, 1!
170 Lær: Ja, 1 ganges med 2, okei da blir det 2
171 Nin: 2 gange 1?
172 Lær: Du kan prøve da, så ser dere hva som skjer da. (De bytter ut 1x2 med 2x1. Stille i noen sekunder mens vi observerer skjermen)
173 Nin: Nå går det oppover
174 Lær: Vent, hva var det som skjedde? Jeg så ikke helt selv
175 Nin: Da ble det 2..4, 6, 8, 10, 12
176 Lær: Ja. (De endrer til 2x2). Nå er det 4, 8, 4, 8. (De trykker på restart igjen før den rekker å gå videre fra 8)

Figur 4.3: Variabelverdi uttrykkes som det symbolske tallet 1

Brage sier først at det er det første tallet som ganges med 2 før han i stedet omtaler verdien 1. På linje 176 kan vi også se at de gikk fra 2x1 til 2x2. Når de bytter ut 1 med 2 som andre faktor, virker intensjonen å være å bytte ut faktoren i takt med det som er det neste tallet i følgen. Elevene demonstrerer at de ønsker en dobling av hvert nye tall. Dette ville blitt oppnådd ved å bruke variabelverdien, slik: «(det første tallet) x 2». Men læringsparet bruker ikke variabelblokka for å uttrykke variabelverdien, i stedet bruker de symbolske tallverdier. Løsningen de ønsker å oppnå ville riktignok ikke fungert med endreblokk, men det de prøver på er «endre (det første tallet) med (det første tallet) x 2». Dette ville ikke gitt riktig tallfølge, men tankegangen med dobling av det første tallet er matematisk korrekt.

I det følgende utdraget inngår Nina, Brage og Fredrik:

145 Nin: Mhh, vi får bare til 1, 2, 3, 4, 5, 6, 7, 8, ...
146 Lær: Men det her ser jo veldig spennende ut da. Jeg skal se litt nærmere på det snart. (De bytter så ut 1 med 2 i endreblokka, får 2, 4, 6, 8, ...)
147 Nin: Vis det første tallet med..Vi må ha, vis det første gangen vi kan, tallet med ganger
148 Bra: Gange 2. Hvor er gangetegnet? (skriver «x2» inn i verdiplassen)
149 Fre: Hvor langt har dere kommet?
150 Nin: Gangetegnet?
151 Fre: Dere har kommet mye lengre enn oss
152 Bra: 2..Det fungerte ikke

Figur 4.4: Forsøk på å gange med 2

På linje 147 foreslår Nina å bruke det første tallet, enten i vis tall eller i endreblokka. Det er noe uklart hva hun mener, men jeg vil trekke fram det siste hun sier, «tallet med ganger». Det kan tenkes at Nina ønsker å uttrykke «(det første tallet) x noe». Hvis dette er tilfelle uttrykker hun behovet for variabelverdien uten at dette følges opp videre, hverken da eller senere.

Et fellestrekk for alle elevene er at variabelen «det første tallet» aldri utforskes som blokk. Det finnes eksempler på uttalelser hvor det første tallet omtales korrekt i en matematisk sammenheng, mens variabelen allikevel ikke blir brukt i programmet, annet enn ved «vis tall».

Astrid og Hanne brukte blokkprogrammet på samme måte i forsøk på å representere tallfølgen. Helt mot slutten av økta, like før det korrekte programmet blir presentert overfor elevene, uttrykker Astrid hvordan tallfølgen fortsetter, med naturlig språk:

410	Lær:	Og så sier dere, at når vi trykker en gang, så skal vi endre det første tallet med 1, men da får dere jo 2. Og etterpå så får dere 3. Og så får dere 4
411	Ast:	Endre det første tallet pluss
412	Lær:	Med pluss hva da? Hva skal vi plusse med?
413	Ast:	1
414	Lær:	1?
415	Ast:	Nei..med..plusse med det samme tallet
416	Lær:	Mhhm, plusse med det samme tallet. Nå har du egentlig sagt det, Astrid. Spørsmålet er bare hvordan vi gjør det innpå der (MakeCode)
417	Ast:	Eh..du må ta pluss..og så legge det til der (hun gjør det selv). Og så..2 pluss 2 er lik 4
418	Lær:	Okei (programmet kjøres, får 1, 5, 9, 13..)
419	Ast:	Det hjalp ikke

Astrid påpeker den rekursive beskrivelsen av følgen, addisjon med det samme tallet. Denne tankegangen samsvarer med fasiten. I forsøk på å uttrykke det hun har uttalt skriver hun inn 2+2 i endreblokka (se figur 4.5). Når man har trykket på knapp A en gang har variabelen verdien 2, men det må ikke forveksles med det symbolske tallet 2 som er uavhengig av variabelens verdi. 2+2 er en addisjon med det samme tallet, men her uttrykkes den gjentatte addisjonen med symbolske verdier ikke med variabelverdier ved hjelp av variabelblokka.

Astrids samtale med lærer bringer oss til neste kategori som omhandler elevenes bruk av

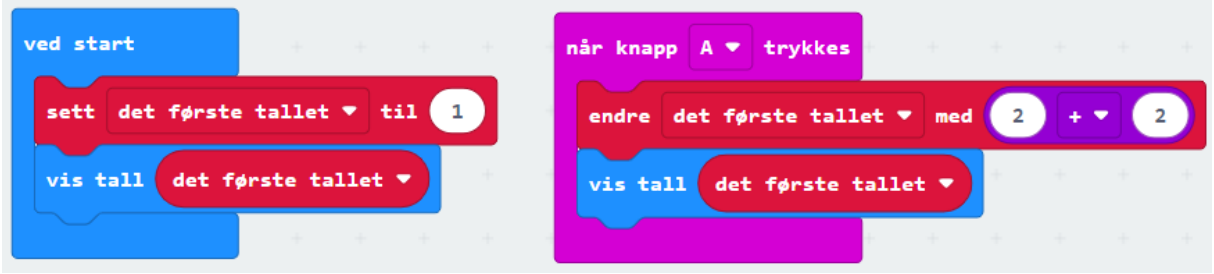
endreblokk.

4.2.2 Elevene benytter endreblokk uten å ta hensyn til addisjonen den betinger

Endreblokk i MakeCode har en innebygd addisjon som er skjult i teksten. «Endre (det første tallet) med 5» betyr at det første tallet øker med 5. Hvis det første tallet er 3 vil en endring med 5 gi verdien 8. Det er også mulig å skrive inn −1, da vil variabelen øke med −1, altså bli én mindre. I repetisjonsfasen, før programmering begynte, viste elevene at de forsto at endreblokken gir en økning. På det tidspunktet hadde de riktignok kun prøvd å sette inn enkeltverdier i endreblokka, ikke operasjonsblokker slik som $1x2$ eller $2+2$. Hvis man benytter «endre (det første tallet) med ($1x2$)» vil det første tallet øke med 2, siden $1x2$ er 2.

Samtale mellom lærer og Astrid ovenfor viser hvordan Astrid skriver inn 2+2 i endreblokka, se også figur 4.5. Astrid ønsker å uttrykke at neste tall i følgen er det nåværende tallet addert med seg selv, derav 2+2. (Den daværende variabelverdien var 2). Ved å skrive inn 2+2 i endreblokka oppnår Astrid at det første tallet øker med 4, ikke med 2 slik hun ønsker. Endreblokka inkluderer alltid variabelnavnet, programmererens oppgave blir å sette inn verdien variabelen skal øke med. Siden endreblokka alltid inkluderer variabelen, uttrykker endreblokka allerede «(det første tallet) + _». Variabelen «det første tallet» hadde verdien 2 på tidspunktet hvor Astrid skrev inn 2+2. Hun behøvde egentlig *bare* å sette inn «det første tallet» i den ledige plassen i endreblokka. Når hun skriver inn 2+2 oppnår hun egentlig regnestykket 2+(2+2), fordi endreblokka allerede inneholder 2+_.

Som beskrevet i 4.2.1, utforskes aldri «det første tallet» som blokk, slik det gjøres med operasjonsblokker og endreblokka. Figur 4.5 nedenfor illustrerer løsningen som Astrid presenterte i forbindelse med at det første tallet må adderes med seg selv.



Figur 4.5: Astrids løsning

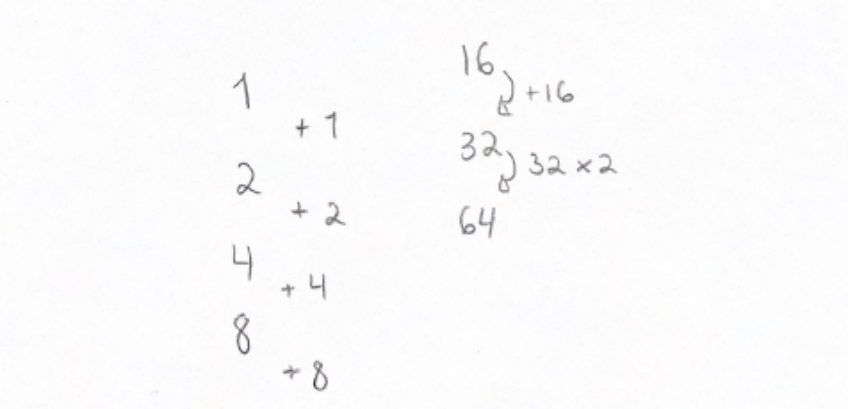
Astrids løsning med å sette en operasjonsblokk inn i endreblokka var noe alle elevene var innom gjentatte ganger.

Under repetisjonen i begynnelsen av økta, sier Nina at hun oppfatter endreblokken som en økning:

41 Lær: Ja, skjønner. Okei, men da kan vi se på D (pil D). Den peker på..
42 Bra: Endre det første tallet med 1
43 Lær: Ja, hva vil det si at man endrer det med 1?
44 Nin: Og ja, det, det! Da blir..da blir det 1, 2, 3, 4, 5, 6, 7, 8..
45 Lær: Er dere enige?
46 Mar: Ja
47 Lær: Da endrer det..Endre, betyr det å øke eller å få en mindre?
48 Nin: Å øke
49 Lær: Å øke med 1? Okei

I prosessen med å representere tallfølgens økning i programmet, uttrykker Nina behovet for pluss. Når Nina uttrykker behovet for pluss, foreslår hun å bruke lilla operasjonsblokk som benytter pluss til å gjøre en operasjon. Denne lillablokken plasseres deretter inn i endreblokken. Denne programvarianten tilsvarer Astrids forslag «endre (det første tallet) med (2+2)». Både Nina og Astrid benytter endreblokken uten å ta hensyn til at den alene utøver addisjon. De andre elevene gjør det samme, og motsier heller ikke Nina og Astrid når de gjør det.

Det er først etter at jeg modellerte tallfølgens fortsettelse at Nina foreslår bruk av pluss. Se figur 4.6 for min modellering på papir.



Figur 4.6: Modellering av følgen som gjentatt addisjon

331 Nin: 1+1 2, 2+2 4, 4+4 8, 8+8 16, 16+16 32
332 Bra: 64
333 Lær: Nå sier du det egentlig. Nå har dere egentlig sagt det
334 Bra: Har vi?
335 Lær: Hvis du har 1, for at det skal bli det neste tallet, må du plusse på 1
336 Bra: Gange
337 Nin: Vi må ha pluss 2. Pluss 2..pluss..Vi må innpå pluss (går innpå matematikk). Pluss.
338 Bra: Men hvordan skal det plusse da?
339 Nin: 1 pluss 1, 1 pluss 1. (Får oddetallsfølge på skjermen)

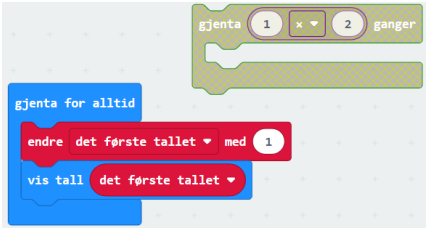
Ninas forslag om å bruke pluss er matematisk sett riktig. I registeret naturlig språk sier hun pluss og velger deretter en blokk som inneholder pluss i blokkprogrammering som register. Om hun skal ta i bruk pluss i blokkprogrammering som register, er en av mulighetene å bruke endreblokken for å endre variabelens startverdi, men da må man ta hensyn til

addisjonen som ligger skjult i blokka.

4.2.3 Elevene bruker løkke for å uttrykke tallfølgens økning

En løkke i programmering sørger for at et bestemt innhold gjentar seg x antall ganger (Istad & Kristoffersen, 2019). Flere ganger setter elevene inn det de anser som dobling ($1x2$) inn i antall ganger løkka skal gjenta seg.

Brage og Nina tar i bruk løkke veldig tidlig i programmeringsprosessen. Gjentatte ganger plasserer de den lilla operasjonsblokken med $1x2$ der hvor man skriver antall ganger løkka skal gjenta seg. Bare noen minutter inn i programmeringsprosessen tipser jeg elevene om å tenke over hva vi gjorde i den første økta, og programmet vi brukte da. Brage og Nina tok deretter i bruk noen av blokkene fra det gamle programmet, se figur 4.7a. Løkka med $1x2$ drar de ut, slik at de kun observerer det nye programmet alene. De får heltallsfølgen {1, 2, 3, 4, 5, ...}. Deretter setter de løkka inn igjen, i håp om at den vil endre tallfølgen med dobling som utgangspunkt, se figur 4.7b.



(a) Løkka tas ut av programmet

(b) Løkka settes inn igjen

Figur 4.7: Program med og uten løkke

144 Nin: Ser du det blir høyere. Men, så nå kan vi ta inn gange. Nå kan vi ta inn gangegreia
145 Nin: Ehm, vi får bare til 1, 2, 3, 4, 5, 6, 7, 8...

Ut fra samtalene forstår vi at de ønsker å fortelle programmet at det skal doble tallene, men det de egentlig gjør, ut fra hvordan løkker fungerer, er å gi beskjed om at innholdet i løkka skal gjenta seg $1x2$ ganger. Altså to ganger. Innholdet i løkka forblir det samme.

På linje 145 oppdager Nina at løkka ikke medfører en dobling av tallene i følgen. Operasjonsblokken fører bare til at løkka gjentar at det første tallet skal endres med 1, to ganger. Siden løkka står inni «gjenta for alltid» vil også sekvensen med å «endre (det første tallet) med 1» og «vis tall (det første tallet)», gjenta seg for alltid. Resultatet av programmet blir en uendelig heltallsfølge.

5 Diskusjon

Basert på funnene jeg presenterte i det foregående kapitlet, vil jeg nå utdype hvordan funnene knyttes til min problemstilling: *hvordan bruker elever på 4. trinn blokkprogrammering som semiotisk representasjonsregister i arbeid med tallfølger*. Å gjøre behandling og omdanning er en del av det å bruke et register (Duval, 2006, 2017). Å bruke et register krever også at man forholder seg til de interne betingelsene som gjelder i registeret (Duval, 2017). Ut ifra disse betraktningene vil jeg diskutere hvordan funnene fra analysen anses å være bruk av blokkprogrammering som register, samtidig som jeg diskuterer muligheter og utfordringer ved blokkprogrammering som register for tallfølger. Avslutningsvis vil jeg diskutere de metodiske valgene i studien som fikk særlig betydning for elevenes matematiske arbeid.

5.1 Bruk av blokkprogrammering som register

Elevene i min studie fikk se en tallfølge som en symbolsk representasjon, altså tilhørende det symbolske registeret. Ved å studere tallfølgen skulle de gjøre en omdanning til blokkprogrammering som register, gjennom å lage et blokkprogram som representerte den samme tallfølgen. Ut ifra Duvals (2006) teori om behandling og omdanning legger oppgaven opp til et registerskifte fra symbolsk register til blokkprogrammering som register. Behandling av representasjoner skjedde i det symbolske registeret når Astrid og Hanne symbolsk representerte hvordan de forsto tallfølgens økning. Se figur 4.1. Behandling innad i blokkprogrammering som register var i utgangspunktet ikke nødvendig, fordi en behandling er en endring av representasjon innad i samme register (Duval, 2004). Hvis elevene hadde mestret omdanning fra symbolsk tallfølge til tallfølge i blokkprogrammering, hadde det ikke vært behov for å gjøre behandling i programmeringsmiljøet, fordi oppgaven hadde vært oppnådd.

Omdanning er mer kognitivt krevende enn behandling (Duval, 2006). Elevene i min studie mestret ikke en fullstendig omdanning fra symbolsk register til blokkprogrammering som register. Elevene gjorde raskt omdanning fra symbolsk register til naturlig språk, og brukte mye tid og innsats på å forsøke videre omdanning fra naturlig språk til blokkprogrammering. Funnene fra analysen viser hvordan tre betingelser ved blokkprogrammering som register, ble til hinder for elevenes omdanning.

Hvordan man uttrykker variabelverdier, skjulte egenskaper ved endreblokka, og hvordan man benytter løkker, er tre betingelser ved blokkprogrammering som elevene ikke imøtekom. Elevenes bruk av blokkene ga alltid fungerende program, fordi blokkenes former forhindrer syntaksfeil (Weintrop & Wilensky, 2017). Resultatet av programmene ble riktignok sjelden slik elevene forventet, fordi de enten ikke var klar over, eller bevisst på, blokkenes betingelser. Svært ofte lagde elevene et program de trodde ville gi tallfølgen {1, 2, 4, 8, 16, ...}, men som i realiteten ga en annen tallfølge. Slik elevene brukte blokkene, var det umulig for dem å oppnå en korrekt representasjon for tallfølgen. Dermed ble de forhindret fra å gjøre omdanning fra den symbolske tallfølgen til tallfølgen representert som blokkprogram.

Mot slutten av den andre økta påpeker Astrid hvordan økningen fra det første tallet som startverdi må være en addisjon med det samme tallet. Med andre ord uttrykker hun en

nesten kongruent omdanning fra tallfølgen som symbolsk representasjon til naturlig språk. En kongruent omdanning er når representasjonen man gjør omdanning *til* sammenfaller fullstendig med representasjonen man gjorde omdanning *fra* (Duval, 2006). Utsagnet er ikke kongruent med den symbolske representasjonen for tallfølgen, men den er nesten kongruent med blokkprogrammet.

411 Ast: Endre det første tallet pluss
412 Lær: Med pluss hva da? Hva skal vi plusse med?
413 Ast: 1
414 Lær: 1?
415 Ast: Nei..med..plusse med det samme tallet

Hvis vi ser Astrids utsagn på linjene 411 og 415 i sammenheng, ser vi at hun uttrykker «endre det første tallet pluss det samme tallet». Blokkprogrammet tilsvarer «endre (det første tallet) med (det første tallet)». Utsagnet til Astrid er ikke fullstendig kongruent, men nesten. Hun omtaler pluss som noe som skjer mellom det første tallet og «det samme tallet». Denne delen av utsagnet samsvarer med det endreblokka gjør. Jeg vil imidlertid ikke definere utsagnet som fullstendig kongruent med programmet, fordi hun omtaler «det samme tallet» der programmet behøver variabelen «det første tallet», på nytt. «Det samme tallet» er ikke et tegn i blokkprogrammet som semiotisk representasjon, Astrids utsagn samsvarer derfor ikke fullstendig med blokkprogrammet. Når hun sier det samme tallet så er det matematisk sett korrekt, fordi det leddet man skal addere med tross alt er det samme tallet, eller den samme verdien. I naturlig språk er det intuitivt å si «det samme tallet», men da må man være bevisst på å bruke variabelen, og ikke uavhengige symbolske tallverdier, som verdi.

En utfordring ved endreblokken, som kan gjøre omdanningen krevende, er hvordan addisjonen i blokka er skjult. Symbolet +, som elevene kjenner fra det symbolske registeret er usynlig i endreblokka, selv om den matematisk sett er til stede. Astrid velger å sette inn en lilla operasjonsblokk som inneholder plusstegnet for å representere det hun sa om gjentatt addisjon. I repetisjonsfasen før programmeringsprosessen uttrykte både Nina og Hanne at endreblokk betydde økning eller pluss. Dette velger samtlige elever like fullt å se bort ifra, og plasserer operasjonsblokker i verdiplassen til endreblokka. Når Astrid representerer utsagnet sitt som blokkprogrammering, setter hun 2+2 i verdiplassen til endreblokka. Astrid ser behovet for pluss og leter opp en matematikkblokk som inneholder symbolet pluss. Hennes forsøk på å uttrykke tallfølgen tilsvarer imidlertid 2+(2+2), ikke bare 2+2, selv om sistnevnte er hva hun setter inn i programmet. I tillegg til at addisjonen er usynlig må man også være bevisst på at det første leddet i addisjonsstykket allerede er inkludert i endreblokka fra før. Hvis variabel per nå har verdien 2 og endreblokka uttrykker «endre det første tallet med (0)», er det første leddet i regnestykket allerede inkludert i regnestykket, 2+0. *Det første tallet* bærer verdien 2, elevens oppgave blir kun å definere det andre leddet i addisjonsstykket, ikke begge.

Ikke-kongruent omdanning er utfordrende for elever som lærer matematikk (Duval, 2006). Omdanning fra en symbolsk representert tallfølge til tallfølge representert som blokkprogram, er en ikke-kongruent omdanning. Astrid og Hannes beskrivelse av tallfølgen på papiret (se figur 4.1), er nært knyttet til hvordan endreblokka fungerer i blokkprogrammet. En fullstendig kongruent omdanning mellom registrene er likevel ikke mulig i dette tilfelle. Jen-

tene skriver 1+1=2, 2+2=4 osv. Denne beskrivelsen med gjentatt addisjon har likhetstrekk med «endre (det første tallet) med (det første tallet)». Jentenes beskrivelse og tankegang er matematisk sett riktig, mens deres beskrivelser er et godt eksempel på hvordan omdanning fra en symbolsk tallfølge til tallfølge som blokkprogram er en ikke-kongruent omdanning. Dette er det flere grunner til.

Den første grunnen er at man må bruke to linjer med blokker for å uttrykke likningen 1+1=2 eller 2+2=4. Endreblokka vil gi svaret på operasjonen den innehar, men svaret på operasjonen er usynlig, med mindre man legger til «vis tall (det første tallet)». Jeg har alltid inkludert «vis tall (det første tallet)» i utformingen av blokkprogrammene brukt i undervisningsopplegget, for å gjøre alle tall synlige for elevene. Beskrivelsen 1+1=2 behøver både «endre (det første tallet) med (det første tallet)» og «vis tall (det første tallet)» for å utgjøre hele likningen.

Den andre grunnen til at omdanning fra symbolsk tallfølge til blokkprogram er ikke-kongruent, er hvordan blokkene i programmet ikke benytter de samme symbolene (+ og =) som i jentenes beskrivelse på papir.

5.2 Konflikter på tvers av registre

Bråting og Kilhamn (2021) presiserer at begrepene variabel, algoritme og likhet har ulike betydninger i programmering og algebra. For min studie er det relevant å diskutere ordet «endre». Som nevnt sa elevene i repetisjonen i den andre økta, at «endre» i programmet betyr pluss eller økning, noe som er riktig. I programmeringsfasen tok elevene riktignok ikke hensyn til denne observasjonen og brukte endreblokka for å gjøre både multiplikasjon og addisjon. «Endre» betyr pluss i blokkprogrammering med MakeCode, men betyr *forandring* i naturlig språk. Forandring kan referere til all slags endring: økning, redusering, forandring av form, farge osv.

Semiotiske representasjoner består av tegn som sammen med andre tegn gir mening til representasjonen (Duval, 2006, 2017). Ordet endre fungerer som et tegn i begge de semiotiske representasjonsregistrene, men har ulike mening. Når programmering tilfører nye betydninger til tegn, og samtidig innebærer en annen syntaks, kan det oppsto utfordringer for elevene (Qian & Lehman, 2017 i Bråting & Kilhamn, 2021). Hvis elevene leser «endre» i programmet, men tenker «forandring», er det kanskje ikke så unaturlig at de velger å prøve ut dobling som multiplikasjon i en blokk som egentlig kun utfører addisjon.

Nina og Brage oppdager selv en konflikt mellom naturlig språk og såvel blokkprogrammering som symbolsk register. Konflikten oppstår omkring tegnet «x». Løkkeeksemplet fra 4.2.3 Elevene bruker løkke for å uttrykke tallfølgens økning, hvor elevene setter inn en løkke i håp om at den skal doble tallene i følgen. I samtaleutdraget nedenfor tar elevene opp betydningen av x i programmet. Programmet i figur 5.1 hører til samtaleutdraget:

153 Nin: Ja, men vi må ha gange
154 Bra: Gange er x 2
155 Nin: Nei det er det ikke. Gange er ikke x 2, det blir noe annet
156 Bra: Nå står x [hører ikke resten]
157 Nin: x ja, men x og gange er ikke det samme
158 Nin: (til lærer:) Vi finner ikke gangetegnet

Figur 5.1: Forsøk på å gange med 2

Elevene ønsket å oppnå at hvert tall ganges med 2. Dette skrev de inn som «x2» i verdiplassen til endreblokka. Som vi ser av figur 5.1 blir feltet merket med rødt, fordi det ikke er en gyldig mulighet i programmet. Blokkenes former vil i utgangspunktet forhindre syntaksfeil (Weintrop & Wilensky, 2017), men i figur 5.1 ser vi eksempel på et elevdefinerte forslag som blir et unntak for denne påstanden. Program hvor bokstaver settes i verdiplasser, vil ikke kjøre. Dette eksemplet er det eneste tilfelle hvor elevene lagde et ikke-fungerende program.

På linje 157 sier Nina at x og gange ikke er det samme. Her er hun inne på noe vesentlig. Den lilla operasjonsblokka de bruker i programmet innholder tegnet x, men x kan ikke fungere i endreblokka når det er skrevet inn manuelt. Denne bruken av x godtas ikke i programmet, fordi kun tall kan stå på denne plassen, og programmet tolker x som en bokstav. X, som i naturlig språk er en bokstav, er i det symbolske registeret en ukjent verdi, eller et gangetegn. I og med at elevene skrev inn x2, kan det tenkes at de tok i bruk sin kjennskap til det symbolske registeret og x som gangetegn. Det kan også tenkes at elevene benyttet x, fordi de så at blokkprogrammet godtar x i lilla operasjonsblokker. Altså ser vi at tegnet x har ulike mening i de tre registrene naturlig språk, symbolsk og blokkprogrammering. Elever i arbeid med matematikk befinner seg gjerne i flere register samtidig (Duval, 2006, 2017). I dette eksemplet kan det se ut som Brage og Nina befinner seg i alle de tre nevnte registrene. Når hvert register har sine egne betydninger av x kan det oppstå konflikt i elevenes forsøk på omdanning.

Til slutt vil jeg diskutere frasen «det første tallet». Frasen har i hovedsak to betydninger i min studie: i naturlig språk betyr det det faktisk første tallet i følgen, i blokkprogrammet betyr det variabelverdien. Jeg ga variabelen dette navnet, fordi det første som skjer i programmet er at det defineres en startverdi. Det første tallet er 1, derfor ble det naturlig å kalle startverdien for «det første tallet». Navnet gir kanskje mest mening før variabelen endrer verdi, men prosessen som skal foregå med å endre startverdien er rekursiv, fordi det neste tallet som kommer er en endring av det forrige (Adams & Essex, 2013). Før den første endringen er startverdien det forrige tallet. «Det første tallet» blir nye tall underveis, slik Hanne påpeker det:

481 Lær: Endre 1 med 1? (peker på endreblokken)
482 Han: Jaa
483 Lær: Hva betyr det?
484 Han: 2
485 Lær: 2 ja. Okei, så da vises det 2 når jeg har trykket en gang
486 Han: Men da må du endre 2 med det første tallet som blir et annet tall. Da blir..da er det første tallet blitt 2. 2+2 og da blir det åt..fire. Nei, jo 4. Så da blir det 4+4 og så bla bla bla bla bla

Uttalelsen «da er det første tallet blitt 2» forteller oss at Hanne er fortrolig med at en variabel, selv med navn «det første tallet», endrer verdi underveis. Kilhamn et al. (2022) påpeker at det i programmeringssammenheng kan føre til færre feil, hvis man definerer variablene med beskrivende ord. Hanne opplever tilsynelatende ikke konflikt mellom frasen i naturlig språk og frasen i blokkprogrammering som register. Imidlertid var det ingen elever som på eget initiativ utforsket variabelblokken. Det var kun når jeg viste fram denne bruken av blokka at elevene poengterte hva den gjorde.

5.3 Metodediskusjon

Min studie har fokusert på elevers arbeid med én tallfølge: {1, 2, 4, 8, 16, ...}, og dermed kun ett blokkprogram. Det hadde vært en styrke for studien om jeg hadde inkludert arbeid med flere tallfølger. Det ville gitt meg et bredere grunnlag for å kommentere elevers bruk av blokkprogrammering i arbeid med tallfølger. Til tross for det begrensede materiale har jeg fått studert 7 elevers arbeid med tallfølgen, og alle elevene programmerte i 30 minutter sammenhengende. Jeg vil derfor påstå at materialet var omfattende når det gjaldt denne ene tallfølgen, men at det hadde vært en styrke og studert elevers arbeid med flere tallfølger.

Elevenes beskrivelse av strukturen i tallfølgen preget programmeringsprosessen. Brage og Nina beskrev hovedsakelig tallfølgen som en dobling ved multiplikasjon med 2. Astrid og Hanne beskrev tallfølgen som gjentatt addisjon. Astrid og Hannes beskrivelse ligner mest på tankegangen som kreves for å lage et fungerende blokkprogram for tallfølgen og er etter min mening den eneste tankegangen som potensielt kunne lede mot en selvstendig og fungerende løsning. Brage og Ninas forslag med multiplikasjon med 2 krever mer erfaring med programmering, for å oppnå et fungerende program, enn elevene hadde. Se presisering i 3.4.3 Valg av blokkprogram.

Tallfølgen muliggjør altså to ulike forståelser, hvor kun en leder mot en gjennomførbar løsning. Dette aspektet kan forstås som en uheldig side av denne tallfølgen i programmeringssammenheng. Mye av grunnen til at jeg valgte tallfølgen {1, 2, 4, 8, 16, ...}, var fordi få tallfølger hadde et tilhørende blokkprogram som ikke ble for avansert for elevene. Programmet kunne ikke være for avansert, men samtidig ikke helt likt programmet i den første økta.

Duval (2017, s. 87-88) betrakter ikke-kongruent omdanning i sammenheng med læreres valg av register i utforming av oppgaver. Min analyse og diskusjon påpeker at omdanningen fra symbolsk tallfølge til blokkprogrammering er ikke-kongruent. Det er også flere tegn i blokkprogrammering som register som er i konflikt med tegn i andre registre. Tatt dette i betraktning ville det trolig vært en styrke for studien og gitt elevene tallfølgeoppgaven i

registeret naturlig språk, i stedet for i det symbolske registeret. Elevene i min studie gikk via naturlig språk i forsøk på omdanning til blokkprogrammering. Ved å gi oppgaven i naturlig språk som register hadde ikke elevene nødvendigvis behøvd å gå via et annet register for å komme til blokkprogrammering. En regnefortelling kunne potensielt lagt til rette for mer samsvar mellom det skriftlige språket i oppgaven og blokkprogrammet. Regnefortellingen kunne vært:

Du skal få ukelønn. På mandag tjener du 1 krone. På tirsdag tjener du 2 kroner. På onsdag tjener du 4 kroner. For hver dag som går øker lønna med det du tjente dagen før. Lag et blokkprogram med variabel «lønn» som viser lønna du har i løpet av hele uka.

Siden regnefortellinger tar i bruk språk kan man velge hvor mye informasjon elevene skal få ettersom hvor erfarne de er i programmering. Man kan for eksempel velge å presisere bruk av variabel, forslå variabelnavn eller eksplisitt presisere lønnas økning, slik jeg har gjort i regnefortellingen over. Å presisere lønnas økning er den presiseringen i regnefortellingen som har mest samsvar med blokkprogrammet, fordi formuleringen har et visst samsvar med endreblokka, «endre lønn med lønn».

Undervisningsopplegget i denne studien følger PRIMM-modellen (Sentance et al., 2019), med sekvenseringen P-R-I-M-I-M. Den andre økta i undervisningsopplegget hadde hovedfokus på skapesteget, selv om det ble gjort noe utforsking av det «gamle» programmet fra de foregående stegene. Jeg har i ettertid reflektert over skapesteget ut ifra elevenes utgangspunkt i programmering. I den andre gjennomføringen med Astrid og Hanne ga jeg elevene lov til å bruke knappetrykk, hvilket gir et program som er veldig likt programmet i den første økta. Når et program er veldig likt det forrige, kan man diskutere om økta egentlig handler om modifisering, ikke skaping. Ingen av elevene i studien mestret omdanning til blokkprogrammering som register, og flere elever sa at de syntes oppgaven var for vanskelig. Å endre et eksisterende program er trolig lettere enn å skape et eget fra bunnen av. Tatt alt dette i betraktning kunne det vært fordelaktig for elevene, og studien, om undervisningsopplegget hadde fulgt stegene på andre måter, for eksempel P-R-I-M-P-R-I-M-M. Altså to runder med å forutse, utforske og modifisere, i stedet for en. Denne kombinasjonen av steg ville gitt elevene to muligheter til å lese program før de begynte å skape program selv. Elevene var i stand til å lese programmet for tallfølgen {1, 2, 4, 8, 16, ...} når jeg viste fram riktig løsning. De demonstrerte dermed at de forsto blokkene i programmet, men imidlertid ikke nok til å kunne lage det på egenhånd.

Designstudier er opptatt av en refleksiv prosess mellom teori og undersøkelse, hvor tilpasninger i undersøkelsene diskuteres som ledd i å videreutvikle teorien (Prediger et al., 2015). Jeg gjorde to justering i mitt undervisningsopplegg. Som nevnt over fikk Astrid og Hanne mulighet til å benytte knapper i programmet. De fikk også anledning til å beskrive tallfølgens mønster, på papir. Jeg innførte denne justeringen, for å bevisstgjøre elevene omkring tallfølgens økning. Den symbolske representasjonen jentene lagde (1+1=2, 2+2=4 ...) samsvarer mer med endreblokka, enn den symbolske tallfølgen gjør. Jeg kan ikke påstå at justeringen med å beskrive tallfølgen på papir, medførte at elevene mestret mer enn om de ikke hadde beskrevet tallfølgen på papir. Men justeringen medførte at elevene representerte tallfølgen på en måte som samsvarer mer med blokkprogrammet. Jeg har tidligere nevnt at omdanningen fra symbolsk tallfølge til blokkprogram er ikke-kongruent. Jentenes beskrivelse på papir representerer en *mer* kongruent omdanning, enn omdanningen jeg i

utgangspunktet planla med {1, 2, 4, 8, 16, ...}. Denne justeringen i undervisningsopplegget medførte altså en interessant betraktning knyttet til hvordan et slikt undervisningsopplegg kan tilrettelegge bedre for omdanning.

6 Avsluttende refleksjoner

Jeg innledet min studie med å påpeke programmeringens inntreden i læreplanen i matematikk. Dagens læreplan introduserte programmering i 2020, men forskningen på programmeringens rolle i matematikkfaget er fortsatt mangelfull (se Forsström & Kaufmann, 2018). I denne studien har jeg anvendt Duvals (2006) teori om semiotiske representasjoner for å forsøke å knytte sammen programmeringsaktiviteter i MakeCode med det matematiske temaet tallfølger. Nærmere bestemt har jeg, med støtte i Bråting & Kilhamn (2021), presentert hvordan blokkprogrammering kan fungere som et semiotisk representasjonsregister, og undersøkt hvordan 7 elever på 4. trinn bruker blokkprogrammering som register for å representere tallfølger. Undervisningsopplegget i studien tok utgangspunkt i PRIMM-modellen som ramme for å studere elevers selvstendige bruk av blokkprogrammering som register. Jeg valgte fire fokuselever for videre analyse og fant at elevene gjorde omdanning fra symbolsk register til naturlig språk og fra blokkprogrammering til naturlig språk. Elevene lyktes ikke med omdanning fra en symbolsk representert tallfølge til den samme tallfølgen representert som et blokkprogram. Elevene brukte enkelte blokker i MakeCode på en måte som ikke imøtekom betingelser ved blokkprogrammering som representasjonsregister. Elevenes bruk av endreblokk, variabelblokk og løkke hindret dem i å gjøre omdanning fra symbolsk register og naturlig språk, til blokkprogrammering som register.

6.1 Didaktiske implikasjoner

Min studie er utviklet i samarbeid med LAB-TEd-prosjektet som forsker på samarbeid mellom universitet og skole. Skolen hvor jeg gjennomførte min datainnsamling ønsket mer kompetanse på programmering i matematikkfaget. Til tross for at datainnsamlingen er spesifikt rettet mot et enkelt trinn på denne ene skolen, og deres utgangspunkt i programmering, kan funnene i studien likevel relateres til andre klasser og trinn. Funnene omhandler omdanning mellom ulike registre og betingelser innad i blokkprogrammering som register. Disse funnene oppsto ut ifra arbeidet til de spesifikke elevene i min studie. Utfordringer og muligheter knyttet til omdanning med blokkprogrammering, er allikevel aktuell informasjon for alle som tar i bruk blokkprogrammering, siden det kan anses som et representasjonsregister for tallfølger.

Elevene i denne studien hadde utfordringer med omdanning til blokkprogrammering som knyttes til elevenes bruk av endreblokk, variabelblokk og løkke. Dette var riktignok også blokker elevene ikke hadde så mye kjennskap til fra tidligere matematikkundervisning. Det er relevant for lærere som underviser i programmering, å kjenne til utfordringer ved blokkprogrammering som register. Elever på andre trinn og andre skoler kan potensielt oppleve de samme utfordringene.

Jeg har pekt på at omdanning fra en symbolsk representert tallfølge til tallfølgen representert i blokkprogrammering, er ikke-kongruent. I tillegg har jeg pekt på konflikter mellom meningsbærende tegn i de ulike registrene, slik som ordet «endre». For å tilrettelegge bedre for elevers omdanning mellom representasjoner, kan lærere ta utgangspunkt i funnene fra min studie. Å ta i bruk andre registre enn hva jeg har gjort i min studie, eller utforme oppgaver som krever behandling før man gjør omdanning, kan potensielt tilrettelegge for mer produktivt matematisk arbeid.

Blokkprogrammering i MakeCode gir også muligheter for matematisk aktivitet. Tallfølger kan representeres i MakeCode, så lenge elevene har tilstrekkelig programmeringserfaring for å mestre den ikke-kongruente omdanningen. Tallfølger representert i blokkprogrammering kan være en inngang til å lære om funksjoner, fordi mange tallfølger kan representeres som funksjonsuttrykk. Blokkprogram for tallfølger uttrykker verdienes relasjon med ord, eksempelvis «endre (det første tallet) med (det første tallet)». Å bygge forståelse for verdienes relasjon til hverandre via ord, kan være en interessant vinkling for å lære elever om funksjonsuttrykk. Elevene i min studie går i 4. klasse, men skal programmerere også i de resterende barneskoleårene. Siden de nå har kjennskap til programmering av tallfølger, kan læreren bygge videre på disse erfaringene, og fokusere på programmering og funksjoner på mellomtrinnet.

6.2 Videre forskning

Det finnes svært lite forskning som anvender Duvals (2006) teori om representasjonsregistre i arbeid med blokkprogrammering. Min studie har pekt på elevers bruk av blokkprogrammering, og særlig på utfordringer knyttet til omdanning til blokkprogrammering som register. Det behøves ytterligere perspektiver på blokkprogrammeringens muligheter og begrensinger i arbeid med tallfølger, men også andre matematiske emner. Ut ifra denne studien hadde det vært interessant å vite mer om hvordan elever bruker blokkprogrammering som register i arbeid med funksjonsuttrykk. Tallfølgen {1, 2, 4, 8, 16, ...} og det tilhørende blokkprogrammet relaterer begge til funksjonsuttrykket $a_n = a_{n-1} + a_{n-1}$. Blokkprogrammering kunne derfor være egnet som et representasjonsregister for funksjoner. I hvilken grad ville elever sett likheter mellom et symbolsk funksjonsuttrykk og et blokkprogram som representerer det samme?

Som nevnt i 5.3 Metodediskusjon, kunne jeg valgt å gi elevene tallfølgeoppgaven i registeret naturlig språk, i stedet for symbolsk register. For å lære mer om blokkprogrammering som register hadde det vært interessant med studier som benytter andre kombinasjoner av registre. I min studie har fokuset vært på omdanning med blokkprogrammering som register, ikke behandling. PRIMM-modellen, med sitt modifiseringssteg, kunne vært en inngang til å gjøre behandling av programmet elevene utforsker, før man eventuelt gjør omdanning.

I min studie er det elevenes arbeid som har vært gjenstand for forskning. Forsström og Kaufmann (2018) etterspør mer sosiokulturell forskning på programmering i matematikkfaget. Blokkprogram som utgangspunkt for matematisk samtale, anser jeg som en interessant sosiokulturell vinkling for å forstå mer om hvordan blokkprogrammering kan knyttes direkte mot matematiske temaer. Elevene i min studie mestret omdanning fra blokkprogrammering til naturlig språk, altså klarte de å lese et program og påpeke hvordan det fungerte. Matematisk samtale, med blokkprogrammering som utgangspunkt, gir en rekke muligheter for å utveksle forståelse av programmering, uten byrden det er å lage et program selv. Slik kan matematisk samtale fungere som en myk introduksjon til å deretter lage egne program.

Referanser

Adams, R., & Essex, C. (2018). *Calculus 1* (Eight edition.). Pearson.

Andersen, R. (2022). Block-based programming and computational thinking in a collaborative setting: A case study of integrating programming into a maths subject. *Acta Didactica Norden*, *16*(4). https://doi.org/10.5617/adno.9169

Austin, J., Baker, H., Ball, T., Devine, J., Finney, J., De Halleux, P., Hodges, S., Moskal, M., & Stockdale, G. (2020). The BBC Micro:Bit: From the U.K. to the World. *Communications of the ACM*, *63*(3), 62–69. https://doi.org/10.1145/3368856

Bada, S. O. (2015). Constructivism learning theory: A paradigm for teaching and learning. *Journal of Research & Method in Education*, *5*(6), 66–70.

Balanskat, A., & Engelhardt, K. (2014). *Computing our Future*. https://doi.org/10.13140/RG.2.1.5029.9048

Ball, T., de Halleux, P., & Moskal, M. (2019). Static TypeScript: An Implementation of a Static Compiler for the TypeScript Language. *Proceedings of the 16th ACM SIGPLAN International Conference on Managed Programming Languages and Runtimes*, 105–116. https://doi.org/10.1145/3357390.3361032

Bell, T., Alexander, J., Freeman, I., & Grimley, M. (2009). Computer science unplugged: School students doing real computing without computers. *The New Zealand Journal of Applied Computing and Information Technology*, *13*(1), 20–29.

Benton, L., Saunders, P., Kalas, I., Hoyles, C., & Noss, R. (2018). Designing for learning mathematics through programming: A case study of pupils engaging with place value. *International journal of child-computer interaction*, *16*, 68–76.

Blanton, M. L. (2008). *Algebra and the elementary classroom : transforming thinking, transforming practice*. Heinemann.

Bocconi, S., Chioccariello, A., & Earp, J. (2018). *The nordic approach to introducing computational thinking and programming in compulsory education*. https://doi.org/10.17471/54007

Bråting, K., & Kilhamn, C. (2021). Exploring the intersection of algebraic and computational thinking. *Mathematical Thinking and Learning*, *23*(2), 170–185. https://doi.org/10.1080/10986065.2020.1779012

Bueie, H. (2019). *Programmering for matematikklærere*. Universitetsforlaget.

Clark, T., Foster, L., Sloan, L., & Bryman, A. (2021). *Bryman's social research methods* (Sixth edition.). Oxford University Press.

Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C., & Woollard, J. (2015). *Computational thinking - a guide for teachers*. Computing At School.

Duval, R. (2004). A crucial issue in mathematics education: The ability to change representation register. *Proc. of the 10th International Conference on Mathematics education*.

Duval, R. (2006). A cognitive analysis of problems of comprehension in a learning of mathematics. *Educational studies in mathematics*, *61*(1-2), 103–131. https://doi.org/10.1007/s10649-006-0400-z

Duval, R. (2017). *Understanding the mathematical way of thinking-The registers of semiotic representations*. Springer.

Ferrara, F., & Sinclair, N. (2016). An early algebra approach to pattern generalisation: Actualising the virtual through words, gestures and toilet paper. *Educational Studies in Mathematics*, *92*, 1–19. https://doi.org/10.1007/s10649-015-9674-3

Forsström, S. E., & Kaufmann, O. T. (2018). A Literature Review Exploring the use of Programming in Mathematics Education. *International Journal of Learning, Teaching and Educational Research*, *17*(12), 18–32. https://doi.org/10.26803/ijlter.17.12.2

Franklin, D., Hill, C., Dwyer, H. A., Hansen, A. K., Iveland, A., & Harlow, D. B. (2016). Initialization in scratch: Seeking knowledge transfer. *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, 217–222. https://doi.org/10.1145/2839509.2844569

Gravemeijer, K., & Cobb, P. (2006). Design research from a learning design perspective. I J. van den Akker, K. Gravemeijer, S. McKenney & N. Nieveen (Red.), *Educational design research* (s. 17–52). Routledge.

Istad, R. M., & Kristoffersen, B. (2019). *Forstå programmering: med Java* (2. utg.). Universitetsforlaget.

Kaufmann, O. T., Stenseth, B., & Holone, H. (2018). Programmering i matematikkundervisningen. I A. Norstein & F. O. Haara (Red.), *Matematikkundervisning i en digital verden* (s. 73–96). Cappelen Damm akademisk.

Kaufmann, O. T., & Stenseth, B. (2021). Programming in mathematics education. *International journal of mathematical education in science and technology*, *52*(7), 1029–1048. https://doi.org/10.1080/0020739X.2020.1736349

Kazi, M. J. (2022). *The Effects of Using Finch Robots to Teach Math on Middle School Students' Math Self-Efficacy and Achievement*.

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. https://doi.org/10.1007/s11858-022-01384-0

Kunnskapsdepartementet. (2017). *Overordnet del - verdier og prinsipper for grunnopplæring*. Fastsatt som forskrift ved kongelig resolusjon. Læreplanverket for Kunnskapsløftet 2020.

Kunnskapsdepartementet. (2019). *Læreplan i matematikk 1.-10. trinn (MAT01-05)*. Fastsatt som forskrift. Læreplanverket for Kunnskapsløftet 2020.

Laurent, M., Crisci, R., Bressoux, P., Chaachoua, H., Nurra, C., de Vries, E., & Tchounikine, P. (2022). Impact of programming on primary mathematics learning. *Learning and Instruction*, *82*, 101667.

Miller, J. (2019). STEM education in the primary years to support mathematical thinking: Using coding to identify mathematical structures and patterns. *ZDM*, *51*(6), 915–927.

Mulligan, J. T., & Mitchelmore, M. C. (2013). Early Awareness of Mathematical Pattern and Structure. I L. D. English & J. T. Mulligan (Red.), *Reconceptualizing Early Mathematics Learning* (s. 29–45). Springer, Dordrecht. https://doi.org/10.1007/978-94-007-6440-8_3

NESH. (2021, 16. desember). *Forskningsetiske retningslinjer for samfunnsvitenskap og humaniora* (5. utg.). De nasjonale forskningsetiske komiteene. https://www.forskningsetikk.no/retningslinjer/hum-sam/forskningsetiske-retningslinjer-for-samfunnsvitenskap-og-humaniora/

Nilssen, V. L. (2012). *Analyse i kvalitative studier: den skrivende forskeren*. Universitetsforlaget.

NOU 2015: 8. (2015). *Fremtidens skole: Fornyelse av fag og kompetanser*. Kunnskapsdepartementet. https://www.regjeringen.no/no/dokumenter/nou-2015-8/id2417001/

Papic, M., & Mulligan, J. (2007). The Growth of Early Mathematical Patterning: An Intervention Study. *Mathematics: Essential Research, Essential Practice*, *2*, 591–600.

Postholm, M. B. (2021). *LAB-TEd: Learning, Assessment and Boundary crossing in Teacher education*. NTNU. https://www.ntnu.edu/ilu/lab-ted

Postholm, M. B., & Jacobsen, D. I. (2018). *Forskningsmetode for masterstudenter i lærerutdanningen*. Cappelen Damm akademisk.

Prediger, S., Gravemeijer, K., & Confrey, J. (2015). Design research with a focus on learning processes: An overview on achievements and challenges. *ZDM*, *47*, 877–891. https://doi.org/10.1007/s11858-015-0722-3

Seneviratne, P. (2018). *Beginning BBC micro:bit: A practical introduction to micro:bit development*. Apress. https://doi.org/10.1007/978-1-4842-3360-3

Sentance, S., Waite, J., & Kallia, M. (2019). Teaching computer programming with PRIMM: a sociocultural perspective. *Computer science education*, *29*(2-3), 136–176. https://doi.org/10.1080/08993408.2019.1608781

Sevik, K. (2016). Programmering i skolen. https://www.udir.no/globalassets/filer/programmering_i_skolen.pdf

Stylianides, G. J. (2008). An analytic framework of reasoning-and-proving. *For the learning of mathematics*, *28*(1), 9–16.

Super:bit. (u.å.). *Utstyr til super:bit*. https://www.superbit.no/utstyr-til-superbit/

Usiskin, Z. (1988). Conceptions of school algebra and uses of variables. *The ideas of algebra, K-12*, *8*, 19.

Utdanningsdirektoratet. (2019, 27. mars). *Algoritmisk tenkning*. https://www.udir.no/kvalitet-og-kompetanse/profesjonsfaglig-digital-kompetanse/algoritmisk-tenkning/

van den Akker, J. (2006). Introducing educational design research. I J. van den Akker, K. Gravemeijer, S. McKenney & N. Nieveen (Red.), *Educational design research* (s. 3–7). Routledge.

van Laar, E., van Deursen, A. J., van Dijk, J. A., & de Haan, J. (2017). The relation between 21st-century skills and digital skills: A systematic literature review. *Computers in Human Behavior*, *72*, 577–588. https://doi.org/10.1016/j.chb.2017.03.010

Weintrop, D., & Wilensky, U. (2017). Comparing Block-Based and Text-Based Programming in High School Computer Science Classrooms. *ACM Transactions on Computing Education*, *18*(1). https://doi.org/10.1145/3089799

Wijns, N., Torbeyns, J., Bakker, M., De Smedt, B., & Verschaffel, L. (2019). Four-year olds' understanding of repeating and growing patterns and its association with early numerical ability. *Early Childhood Research Quarterly*, *49*, 152–163. https://doi.org/10.1016/j.ecresq.2019.06.004

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33–35.

Wohl, B., Porter, B., & Clinch, S. (2015). Teaching Computer Science to 5-7 year-olds: An initial study with Scratch, Cubelets and unplugged computing. *Proceedings of the workshop in primary and secondary computing education*, 55–60.

Wu, M. L., & Richards, K. (2011). Facilitating Computational Thinking through Game Design. I M. Chang, W.-Y. Hwang, M.-P. Chen & W. Müller (Red.), *Edutainment Technologies. Educational Games and Virtual Reality/Augmented Reality Applications* (s. 220–227). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-23456-9_39

Vedlegg

Vedlegg 1: Skriveramme for programanalyse

Vedlegg 2: Tankekart fra analyseprosessen

Vedlegg 3: Oppgaveark A1

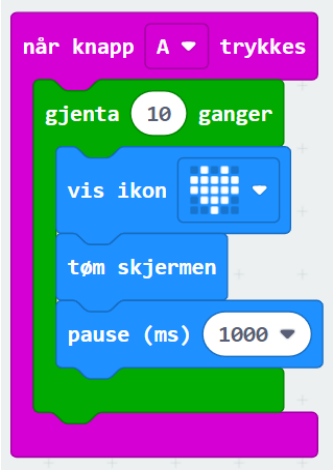
Vedlegg 4: Oppgaveark A2

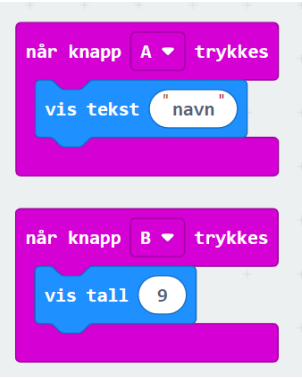
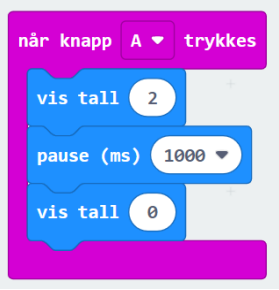
Vedlegg 5: Oppgaveark B2

Vedlegg 6: Samtykkeskjema

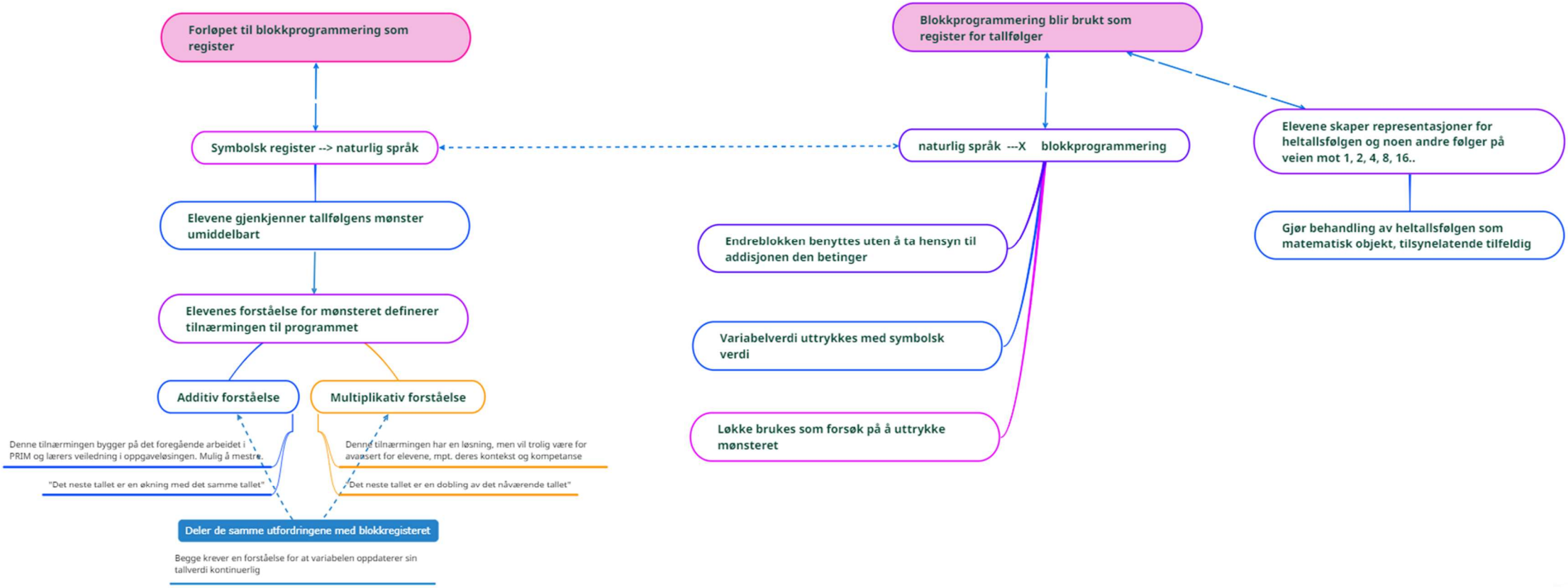
Vedlegg 7: Vurdering fra Sikt

Vedlegg 1: Skriveramme for programanalyse

Kode	Beskriv tydelig og nøyaktig hva koden skal få micro:bit'n til å gjøre	Tegn hva som kommer til å vises på skjermen (micro:bit'n)
		
		

Kode	Beskriv tydelig og nøyaktig hva koden skal få micro:bit'n til å gjøre	Tegn hva som kommer til å vises på skjermen (micro:bit'n)
		
		Hva vises på skjermen hvis temperaturen er 5 (grader Celsius)? Hva vises på skjermen hvis temperaturen er 9 (grader Celsius)?
	Du ønsker at tallet 2 skal vises på skjermen når man trykker på knapp B. Etter en pause på 1 sekund skal tallet 4 vises på skjermen. Finn to feil med denne koden.	

Vedlegg 2: Tankekart fra analyseprosessen



NAVN:

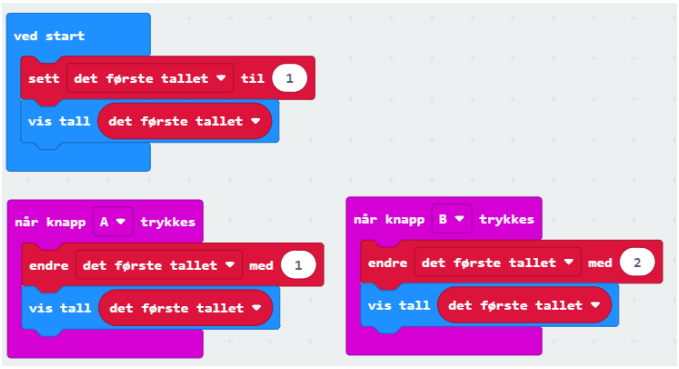
Vedlegg 3: Oppgaveark A1

Mandag 09.01.23

10.30-12.00

1. Se på denne tallfølgen: 1, 2, 4, 5, 7, 8 … Hva blir de tre neste tallene i følgen? Forklar eller vis hvorfor.

2. Se på koden under. Skriv hva du tror den gjør. Hva vil vises på Micro:bit'n sin skjerm?



NAVN:

3. Kjør koden som dere får som lenke. Prøv å trykk på de ulike knappene og se hva som skjer.

4. Skjedde det som du forventet? Viste skjermen det samme som du trodde?

5. Nå skal du få endre på koden selv.

    a. Endre koden slik at det første tallet i tallfølgen blir 5.

Hva tror du tallfølgen blir om du nå trykker A, B, A, B…?

Hva viste Micro:bit'n når du trykket A, B, A, B…?

    b. Endre koden slik at du ved hjelp av knappene får tallfølgen 5, 6, 8, 10, 11, 13…

    c. Endre koden slik at du ved hjelp av knappene får tallfølgen 5, 7, 8, 13, 15, 16, 21..

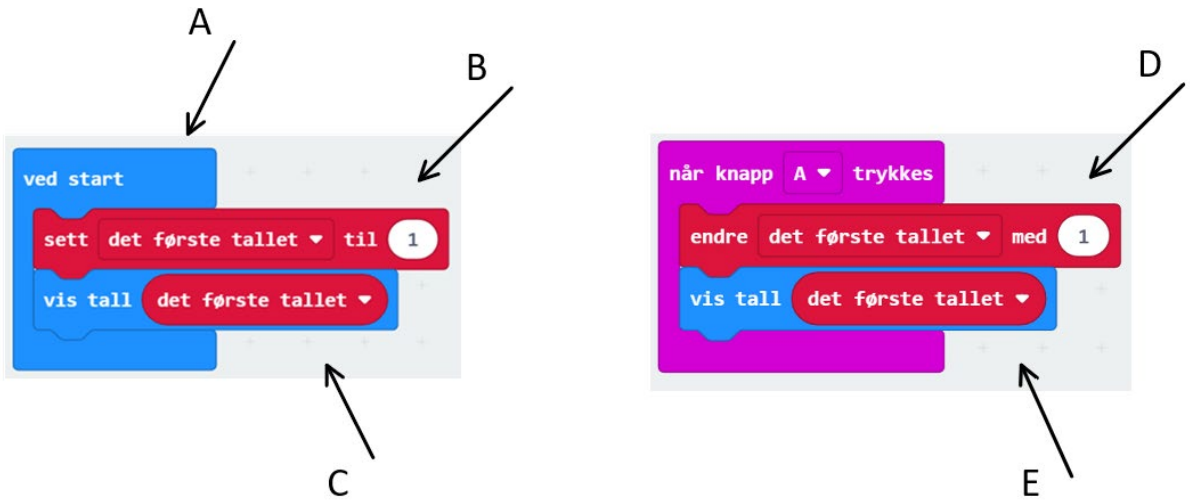
    d. Endre koden slik at du får en tallfølge som du selv vil ha.

Vedlegg 4: Oppgaveark A2

Torsdag 12.01.23

10.30-12.00

Repetisjon fra mandag:

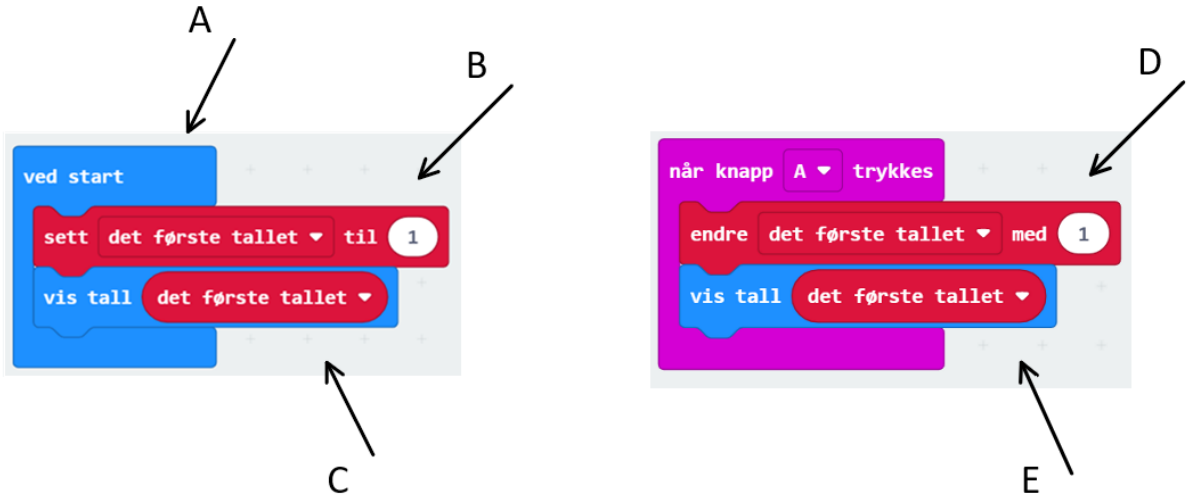


Vedlegg 5: Oppgaveark B2

Uke 4 del 2

10.30-12.00

Repetisjon fra mandag:



Tallfølge:

Vis og forklar hvordan tallfølgen øker:

Vedlegg 6: Samtykkeskjema

Vil du delta i forskningsprosjektet

"Bruk av micro:bit i matematikkundervisning på barneskolen"?

Dette er et spørsmål til deg, som foresatt, om barnet ditt kan delta i et forskningsprosjekt hvor formålet er å undersøke ulike undervisningsopplegg i matematikk som bruker micro:bit. micro:bit er et programmeringsverktøy spesielt rettet mot barnetrinnet. I dette skrivet gir vi deg informasjon om målene for prosjektet og hva deltakelse vil innebære for deg.

Formål
Vi er fire 5.-årsstudenter på Grunnskolelærerutdanningen på NTNU som skal skrive masteroppgaver våren 2023. I den anledning ønsker vi å samle inn datamateriale for å forske på bruk av programmering i matematikkfaget på barneskolen. Innsamlet datamateriale vil være grunnlag for masteroppgavene våre, og alle personopplysninger vil bli anonymisert. Denne forskningen innebærer observasjon, intervju, samt video- og lydopptak av elevene og læreren i klassen.

Vi ønsker å undersøke nærmere:
- Hvordan bruke micro:bit i matematikkundervisning?
- Hvordan kan matematikk inngå i programmering?
- Kan bruk av programmering i matematikkundervisning bidra til økt motivasjon for matematikkfaget generelt?

Masteroppgavene blir veiledet som en del av arbeid i forskningsprosjektet LAB-Ted.
Det anonymiserte datamaterialet innsamlet til masteroppgavene kan inngå i annen forskning knyttet til LAB-Ted-prosjektet.

Hvem er ansvarlig for forskningsprosjektet?
Norges teknisk-naturvitenskapelige universitet er ansvarlig for prosjektet.

Hvorfor får du spørsmål om å delta?
Elever som har [redacted] som matematikklærer ved 4. trinn ved [redacted] får tilbud om å delta i prosjektet.

Hva innebærer det for deg å delta?
Eleven vil delta i den planlagte matematikkundervisningen. I enkelte økter, der micro:bit blir brukt, vil det bli tatt video- og lydopptak av elevenes arbeid. Dette bli lagret på en kryptert, ekstern harddisk, og vil slettes etter at materialet er transkribert og anonymisert.

I disse øktene vil vi snakke med elevene om deres tanker og arbeid med micro:bit. Dette kan enten foregå underveis i klasserommet, eller dersom det er praktisk umulig, vil vi be elevene om en samtale på grupperom etter endt økt. Potensielle spørsmål kan være:
- Hvordan synes du det fungerer å bruke micro:bit i undervisningen?
- Hvordan tenkte du i løsningen av denne oppgaven?
- Synes du programmering er en motiverende arbeidsform i matematikkfaget?
- Hvordan brukte du kodebiten i løsningen din?
- Syns du det er noe som er vanskelig med bruk av micro:bit?

Hvis du som foresatt ønsker en fullstendig intervjuguide, ta direkte kontakt med oss.

Det er frivillig å delta
Det er frivillig å delta i prosjektet. Hvis du velger å delta, kan du når som helst trekke samtykket tilbake uten å oppgi noen grunn. Alle dine personopplysninger vil da bli slettet. Det vil ikke ha noen negative konsekvenser for deg hvis du ikke vil delta eller senere velger å trekke deg.

Elevene vil også bli spurt om de ønsker å delta i forskningsprosjektet, og elever som ikke ønsker dette vil ikke delta. Alle elevene vil få samme undervisningsopplegg, men det vil ikke bli samlet inn datamateriale av elever som ikke ønsker å delta.

Ditt personvern – hvordan vi oppbevarer og bruker dine opplysninger
Vi vil bare bruke opplysningene om deg til formålene vi har fortalt om i dette skrivet. Vi behandler opplysningene konfidensielt og i samsvar med personvernregelverket. Personer som vil ha tilgang til opplysningene som blir samlet inn er: studentgruppa, praksislærer og veiledere fra NTNU. Alle personopplysninger vil bli anonymisert, og datamaterialet vil bli lagret på en innelåst og kryptert ekstern harddisk.

Hva skjer med personopplysningene dine når forskningsprosjektet avsluttes?
Prosjektet vil etter planen avsluttes/oppgavene blir godkjent, noe som etter planen er juni 2023. Alle former for opptak som blir gjort under gjennomføring av prosjektet, vil anonymiseres fortløpende og selve opptakene vil slettes senest ved prosjektslutt.

Siden dette prosjektet er en del av et større prosjekt (LAB-Ted), vil de anonymiserte data fra prosjektet bli beholdt noe utover prosjektperioden, men vil slettes senest ved utgangen av juni 2025. Temaet for prosjektet er høyaktuelt for utviklingsarbeid og forskning i norsk skole, og derfor vil det anonymiserte materialet bli beholdt noe utover perioden masterprosjektene forgår. Formålet med den videre lagringen er at forskere ved NTNU, ILU, spesifikt Benedikte Grimeland, Yvonne Grimeland, Oda Tingstad Burhheim og Torunn Klemp kan bearbeide materialet videre for å publisere forskningsarbeid slik at resultatene fra prosjektet når en større del av lærerutdanningsfeltet i Norge. Det er kun de nevnte forskere som vil ha tilgang til materialet. Materialet vil bli lagret på kryptert disk som ikke er koblet til nett, innelåst hos NTNU, og eid av NTNU.

Hva gir oss rett til å behandle personopplysninger om deg?
Vi behandler opplysninger om deg basert på ditt samtykke.

På oppdrag fra NTNU har Personverntjenester vurdert at behandlingen av personopplysninger i dette prosjektet er i samsvar med personvernregelverket.

Dine rettigheter
Så lenge du kan identifiseres i datamaterialet, har du rett til:
- innsyn i hvilke opplysninger vi behandler om deg, og å få utlevert en kopi av opplysningene
- å få rettet opplysninger om deg som er feil eller misvisende
- å få slettet personopplysninger om deg
- å sende klage til Datatilsynet om behandlingen av dine personopplysninger

Hvis du har spørsmål til studien, eller ønsker å vite mer om eller benytte deg av dine rettigheter, ta kontakt med:
- NTNU ved Yvonne Grimeland. Tlf. 48114352. Epost: Yvonne.grimeland@ntnu.no
- Vårt personvernombud: Thomas Helgesen. Tlf. 93079038. Epost: thomas.helgesen@ntnu.no

Hvis du har spørsmål knyttet til Personverntjenester sin vurdering av prosjektet, kan du ta kontakt med:

- Personverntjenester på epost (personverntjenester@sikt.no) eller på telefon: 53 21 15 00.

Med vennlig hilsen

Guro Kjeldstad (forsker) Vilde Dahle (forsker) Ellen-Marie Kristiansen (forsker) Eirun Flaten Sandvin (forsker)

Yvonne Grimeland (veileder)

Samtykkeerklæring

Jeg har mottatt og forstått informasjon om prosjektet «Bruk av micro:bit i matematikkundervisning på barneskolen», og har fått anledning til å stille spørsmål.

Jeg samtykker til at mitt barn har tillatelse til:

- ☐ å delta i intervju
- ☐ å delta i lyd-opptak
- ☐ å delta i video-opptak

Jeg samtykker til at opplysninger om mitt barn behandles frem til prosjektet er avsluttet

---

(Signert av prosjektdeltakers foresatt, dato)

Vedlegg 7: Vurdering fra Sikt

Vurdering av behandling av personopplysninger

Referansenummer	Vurderingstype	Dato
194418	Standard	17.01.2023

Prosjekttittel
Bruk av Micro Bit i matematikkundervisning på barneskolen

Behandlingsansvarlig institusjon
Norges teknisk-naturvitenskapelige universitet / Fakultet for samfunns- og utdanningsvitenskap (SU) / Institutt for lærerutdanning

Prosjektansvarlig
Yvonne Grimeland

Student
Guro Kjeldstad

Prosjektperiode
01.01.2023 - 01.06.2023

Kategorier personopplysninger
Alminnelige

Lovlig grunnlag
Samtykke (Personvernforordningen art. 6 nr. 1 bokstav a)

Behandlingen av personopplysningene er lovlig så fremt den gjennomføres som oppgitt i meldeskjemaet. Det lovlige grunnlaget gjelder til 01.06.2023.

Meldeskjema

Kommentar
OM VURDERINGEN
Sikt har en avtale med institusjonen du forsker eller studerer ved. Denne avtalen innebærer at vi skal gi deg råd slik at behandlingen av personopplysninger i prosjektet ditt er lovlig etter personvernregelverket.

UTDYPENDE OM LOVLIG GRUNNLAG
Utvalg 1: Prosjektet vil innhente samtykke fra foresatte til behandlingen av personopplysninger om barna. Vår vurdering er at prosjektet legger opp til et samtykke i samsvar med kravene i art. 4 og 7, ved at det er en frivillig, spesifikk, informert og utvetydig bekreftelse som kan dokumenteres, og som den registrerte/foresatte kan trekke tilbake.

Utvalg 2: Prosjektet vil innhente samtykke fra de registrerte til behandlingen av personopplysninger. Vår vurdering er at prosjektet legger opp til et samtykke i samsvar med kravene i art. 4 og 7, ved at det er en frivillig, spesifikk, informert og utvetydig bekreftelse som kan dokumenteres, og som den registrerte kan trekke tilbake. Lovlig grunnlag for behandlingen vil dermed være den registrertes samtykke, jf. personvernforordningen art. 6 nr. 1 bokstav a.

ENDRING I MELDESKJEMA 17.01.2023.
Vi viser til endring registrert i meldeskjemaet. Vi kan ikke se at det er gjort noen oppdateringer i meldeskjemaet eller vedlegg som har innvirkning på vår vurdering av hvordan personopplysninger behandles i prosjektet.

DELE PROSJEKTET MED PROSJEKTANSVARLIG
Du må dele prosjektet med prosjektansvarlig. Velg "Del prosjekt" øverst i meldeskjemaet. Hvis prosjektansvarlig ikke godtar invitasjonen innen én uke, må du sende en ny invitasjon.

DE REGISTRERTES RETTIGHETER
Prosjektet vil gjøre tiltak for å ivareta de registrertes rettigheter etter personvernforordningen (art. 12 nr. 1 og 2), og gi informasjon i samsvar med art. 13/14.

De registrerte har i utgangspunktet rett til innsyn, retting, sletting av sine opplysninger, hvis de sikkert kan identifiseres i datamaterialet. Hvis en registrert tar kontakt om sine rettigheter, anbefaler vi at du rådfører deg med institusjonen din så snart som mulig, for bistand. Institusjonen har plikt til å vurdere om rettighetene skal/kan innfris, og svare den registrerte innen en måned. Personverntjenester kan også kontaktes for råd og veiledning om rettigheter.

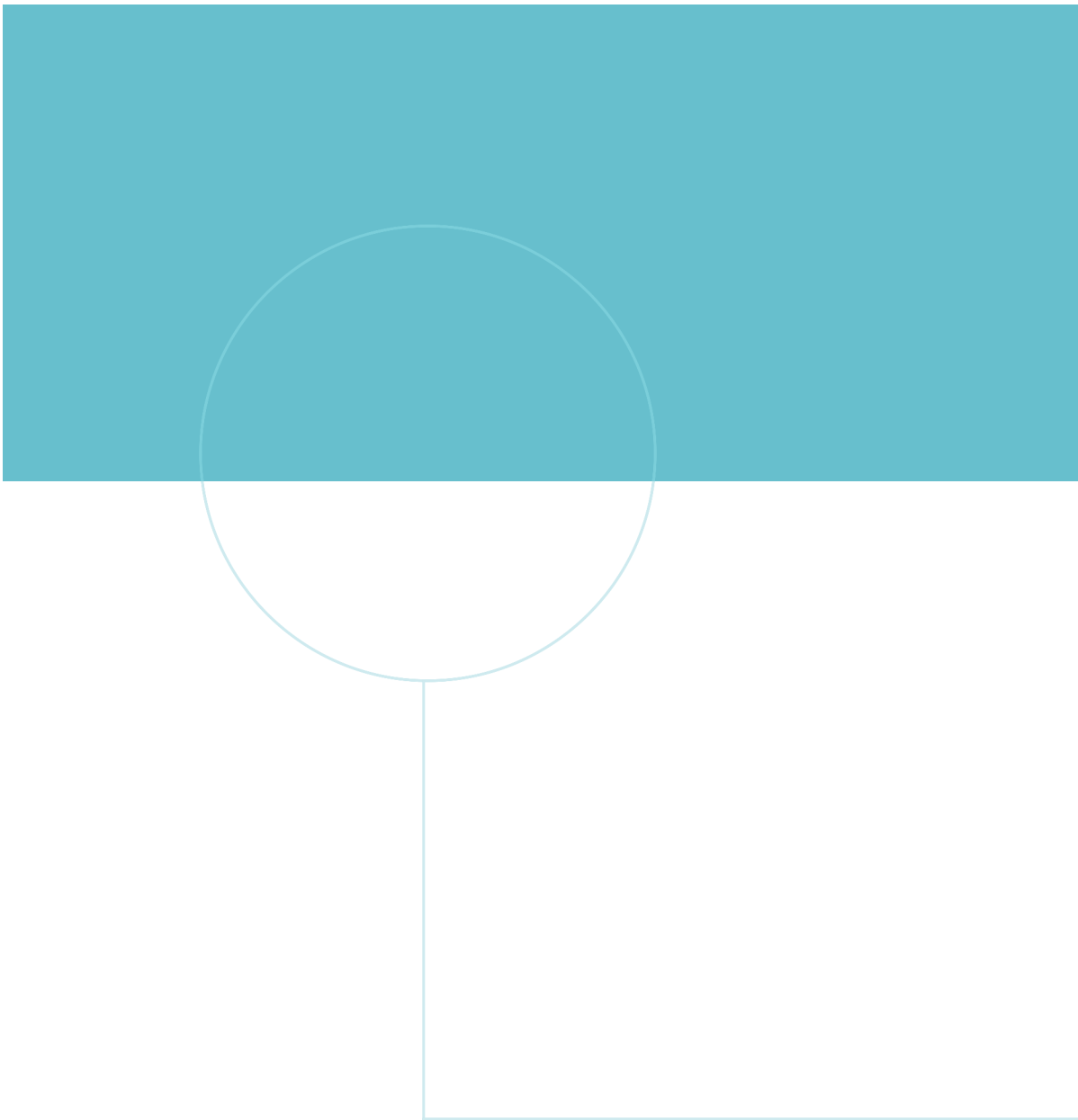
FØLG DIN INSTITUSJONS RETNINGSLINJER
Vi har vurdert at du har lovlig grunnlag til å behandle personopplysningene, men husk at det er institusjonen du er ansatt/student ved som avgjør hvilke databehandlere du kan bruke og hvordan du må lagre og sikre data i ditt prosjekt. Husk å bruke leverandører som din institusjon har avtale med (f.eks. ved skylagring, nettspørreskjema, videosamtale el.)

Personverntjenester legger til grunn at behandlingen oppfyller kravene i personvernforordningen om riktighet (art. 5.1 d), integritet og konfidensialitet (art. 5.1. f) og sikkerhet (art. 32).

MELD VESENTLIGE ENDRINGER
Dersom det skjer vesentlige endringer i behandlingen av personopplysninger, kan det være nødvendig å melde dette til oss ved å oppdatere meldeskjemaet. Se våre nettsider om hvilke endringer du må melde: https://sikt.no/melde-endringar-i-meldeskjema

OPPFØLGING AV PROSJEKTET
Vi vil følge opp ved planlagt avslutning for å avklare om behandlingen av personopplysningene er avsluttet.

Lykke til med prosjektet!



NTNU

Kunnskap for en bedre verden