Programmera sig till matematik?

En analys av programmeringsappars lämplighet för matematikundervisning i årskurs 1-3

Program your way to mathematics?

An analysis of programming apps' suitability for mathematics education in primary school

Sofia Helgen

Fakulteten för hälsa, natur- och teknikvetenskap

Matematik / Grundlärarprogrammet: Förskoleklass och grundskolans år 1-3

Avancerad nivå / 30 hp

Handledare: Maria Fahlgren

Examinator: Yvonne Liljekvist

2018-06-08

Abstract

Starting in the fall of 2018, programming is a part of the Swedish curriculum for mathematics and has been placed under the core content for algebra. Even though it is not obligatory until the fall, some teachers have already started teaching about programming using apps as a digital learning resource. How well do the programming apps fit with the curriculum for mathematics, and how suitable are they to be used in primary school education? With these questions in mind an analysis was made of 14 programming apps for children 6-10 years old. The study is a qualitative content analysis with an analysis model based on the curriculum for mathematics, theories about pre-algebra and criteria that teachers believe to be important in the choice of a good app. The study shows that the apps' suitability for mathematics education vary, but it is the apps that are designed as games that fit best with the curriculum for mathematics. These apps also have the pedagogical design that suits best for primary school education. The study's results question the choice of placing programming under algebra in the curriculum, since the connection between the areas is not quite clear.

Keywords

Apps, digital learning resources, mathematics education, pre-algebra, primary school, programming.

Sammanfattning

Från och med höstterminen 2018 är programmering en del av kursplanen i matematik, och har då placerats under det centrala innehållet för algebra. Trots att det inte är obligatoriskt förrän höstterminen har lärare redan börjat undervisa i programmering, och appar är då en populär lärresurs. Hur väl stämmer apparna för programmering överens med kursplanen i matematik, och hur väl lämpar de sig för undervisning på lågstadiet? Med utgångspunkt i dessa frågor gjordes en analys av 14 programmeringsappar som riktar sig till lågstadieåldrarna, 6-10 år. Studien är en kvalitativ innehållsanalys med en analysmodell som baserats på kursplanen i matematik, teorier om pre-algebra samt kriterier som lärare anser viktiga för en bra app. Studien visar att apparnas lämplighet för matematikundervisning varierar, men att det är de appar som är utformade som spel som stämmer bäst överens med kursplanen i matematik. Det är också dessa appar som har en pedagogisk utformning som lämpar sig bäst för undervisning på lågstadiet. Resultatet av studien ifrågasätter dock valet att placera programmering under algebra i läroplanen, då kopplingen däremellan inte är helt tydlig.

Nyckelord

Appar, digitala lärresurser, lågstadiet, matematikundervisning, pre-algebra, programmering.

Innehållsförteckning

1 Introduktion

Under 2017 beslutade regeringen om en digitaliseringsstrategi för att Sverige fortsatt ska vara ett ledande land när det gäller digital kompetens. Man anser att skolan spelar en central roll i frågan, och en del i strategin blev därför en revidering utav läroplanerna som ska användas i undervisning från och med höstterminen 2018 (Utbildningsdepartementet, 2017). I *Läroplanen för grundskolan, förskoleklassen och fritidshemmet* (Skolverket, 2017b) har vikten av att eleverna ska utveckla digital kompetens fått ta en allt större plats, och som ett led i det har man inkluderat programmering i kursplanerna för matematik och teknik. Arbetet med programmering ska följa genom hela grundskolan, med start redan på lågstadiet (Skolverket, 2017b). Att programmera är att skapa, modifiera eller kombinera datorprogram. Arbetet kräver att man som programmerare är analytisk och systematisk för att kunna upptäcka problem, och att logiskt kunna strukturera upp problemen och deras lösningar så dessa kan förstås av en dator. Programmeraren behöver också vara kreativ och skapande för att kunna designa lösningar och konstruera koder. Kod är den sammansättning av instruktioner som ges till datorn i programmeringen (Helenius, Misfeldt, Rolandsson, & Ryan, 2017).

Trots att programmering i undervisningen inte börjar gälla förrän läsåret 2018/2019 finns indikationer på att flertalet lärare redan börjat undervisa om det. Bland annat har detta blivit synligt på sociala medier såsom Facebook, där lärare ber om förslag på hur de kan arbeta med programmering i sina klasser. Diskussionerna i de sociala medierna visar att många lärare startar med ”unplugged programmering”, det vill säga att eleverna först får öva programmering med fysiska övningar för att lära sig grunderna och att förstå programmeringstänket (Mannila, 2017; Nygårds, 2015). När grunderna har satts, går man vidare till att använda digitala verktyg såsom datorer och surfplattor i undervisningen. Där finns ett flertal digitala lärresurser där eleverna lär sig att programmera och kan skapa egna animationer och spel (Mannila, 2017; Nygårds, 2015). Digitala lärresurser är material som används vid undervisning och lärande som är digitalt. Allt lektionsmaterial som är digitalt ses som en digital lärresurs, vilket innebär att det kan gälla både stora saker som hemsidor och applikationer (appar), och små saker som en bild eller en ljudfil (Skolverket, 2016a). Digitala lärresurser som används för programmeringsundervisning är bland annat appar (Mannila, 2017; Nygårds, 2015).

Programmering må vara ett nytt inslag i grundskolans läroplan, men som tidigare nämnts bedrivs undervisning i det redan, och det finns en del appar att tillgå. Vid valet av app krävs det att läraren besitter pedagogisk och ämnesdidaktisk kompetens, men också digital kompetens (Mishra & Koehler, 2006). I Skolverkets rapport om IT-användning i skolan uppgav ungefär hälften av alla grundskolelärare att de upplevde att det fanns ett kompetensutvecklingsbehov inom kodning och programmering (Skolverket, 2016b), något som även ett flertal remissinstanser uppmärksammade vid utredningen inför de nya tilläggen om programmering och digital kompetens i grundskolans läroplan (Skolverket, 2016c). Det är viktigt att ha i åtanke vid val av app att det inte finns några restriktioner gällande vem som får skapa eller sälja en app för undervisning, och att det inte heller finns något som granskar appens innehåll eller lämplighet (Palmér & Helenius, 2016). Det är alltså upp till lärarna själva att kontrollera apparna. Saknas kompetensen i ämnet, såsom många lärare upplever att det gör gällande programmering, kan det vara svårt att välja ut en bra app som lämpar sig för elevernas nivå.

Som tidigare nämnts har programmering lagts till i kursplanen för matematik i 2017 års reviderade version av *Läroplan för grundskolan, förskoleklassen och fritidshemmet* (2017b), med obligatoriskt tillträde höstterminen 2018. Detta till trots har en del lärare redan plockat in programmering i sin undervisning, och appar är då en populär lärresurs. Lever de appar som riktar sig till programmeringsundervisning på lågstadiet upp till de skrivelser som finns i sagda kursplan?

1.1 Syfte

Syftet med studien är att analysera huruvida de applikationer (appar) som riktar sig mot programmering för barn i lågstadieåldrarna stämmer överens med kursplanen i matematik, samt hur väl de lämpar sig för lågstadiets matematikundervisning.

1.1.1 Frågeställningar

Utifrån studiens syfte har följande frågeställningar valts

- Hur väl stämmer programmeringsapparna överens med kursplanen för matematik?
- Hur väl lämpar sig apparna för undervisning på lågstadiet?

2 Bakgrund: Forsknings- och litteraturgenomgång

I följande avsnitt kommer forskning och litteratur som är relevant för studien att redogöras. Inledningsvis förklaras några begrepp som är centrala i denna studie och för programmering. Därefter redogörs vad som står om programmering i styrdokumenten, varför programmering blivit ett inslag i grundskolans läroplan, samt hur programmering passar in i kursplanen för matematik. I sagda kursplan har programmering placerats inom ämnesområdet algebra; därför kommer det avslutningsvis att redovisas vad algebra innebär på lågstadiet och hur det kan kopplas till programmering.

2.1 Centrala begrepp

2.1.1 Centrala begrepp för studien

Digitala verktyg är ett samlingsnamn för tekniska verktyg som används som hjälpmedel i undervisning. Det innefattar både hårdvaran och mjukvaran. Andra namn som ofta används är IT eller IKT, informations- och kommunikationsteknik (Skolinspektionen, 2011). Digitala verktyg är dock den benämning som används i Lgr11 (Skolverket, 2017b), och är därför den benämning som används även i denna studie.

Ett annat begrepp som är centralt i studien är *applikationer*. Applikationer är en typ av dataprogram, mjukvara, som kallas tillämpningsprogram. Dessa ger den mobila enheten fler funktioner, och används direkt utav användaren till skillnad från exempelvis systemprogram som sköter enhetens inre arbete. Applikationer kan enkelt laddas ner och installeras av användaren. En vanligt förekommande förkortning för applikation är ”app” (plural ”appar”) (Nationalencyklopedin, 2018). Det är förkortningen ”appar” som fortsättningsvis används i studien.

Det tredje och sista centrala begreppet är *digital kompetens*. I takt med att vårt samhälle blir allt mer digitaliserat, är digital kompetens ett begrepp i ständig förändring och därmed svårdefinierat. Digitaliseringskommissionen (2015) beskriver dock digital kompetens som att man är förtrogen med digitala verktyg och tjänster, att man kan söka information och kommunicera digitalt, samt att man har förståelse för hur tekniken påverkar samhället, och att man är motiverad till att delta i utvecklingen.

2.1.2 Centrala begrepp inom programmering

Det finns olika sätt att programmera på. Det sätt som är vanligast på lågstadiet är att programmera med *blockbaserad programmering*. I den blockbaserade programmeringen skriver man inte programmeringen själv, istället finns det färdiga block markerade med korta texter eller symboler som beskriver vad blocken gör (till exempel ”gå framåt”, ”sväng vänster”, ”upprepa 5 gånger”). Blocken sätts sedan ihop till en programmering (Helenius m.fl., 2017; Mannila, 2017). Ett annat sätt att programmera är med *textbaserad programmering* där man själv skriver instruktioner och kommandon i form av text (Helenius m.fl., 2017; Mannila, 2017). Hur programmeringen sätts ihop ser olika ut i olika *programspråk*. Vilket programspråk man väljer beror på vad det är man vill programmera, eftersom språken används för att bygga olika funktioner (Nygårds, 2015). Ibland blir det fel när man skriver programmering; en robot kan svänga åt fel håll eller ett spel kan krascha. Fel i programmeringen kallas för *buggar*, och processen för att hitta felen kallas för *felsökning* (Mannila, 2017).

En programmering består utav olika kommandon och instruktioner som kallas för *satser*. När satser i tur och ordning sätts ihop till en programmering kallas det för *sekvens* (Mannila, 2017). Om en sats repeteras ett visst antal gånger kallas det för *slinga* eller *loop* (Helenius m.fl., 2017; Mannila, 2017). En sats kan också vara en *villkorssats*. Det innebär att det finns flera olika satser att välja mellan i sekvensen, och att en sats bara genomförs om ett villkor är uppfyllt. Till exempel ”om hinder, hoppa”. Finns inget hinder kommer figuren inte att hoppa, utan följer då instruktionerna i en annan sats istället (Helenius m.fl., 2017; Mannila, 2017).

2.2 Programmering i styrdokumenten

Under 2017 reviderades *Läroplanen för grundskolan, förskoleklassen och fritidshemmet,* Lgr11, för att ge större utrymme åt användandet av digitala verktyg, och förtydliga skolans uppdrag att hjälpa eleverna att utveckla och stärka deras digitala kompetens (Skolverket, 2017b). I läroplanens första del om skolans värdegrund och uppdrag kan man läsa:

> Skolan ska bidra till att eleverna utvecklar förståelse för hur digitaliseringen påverkar individen och samhällets utveckling. Alla elever ska ges möjlighet att utveckla sin förmåga att använda digital teknik. De ska även ges möjlighet att utveckla ett kritiskt och ansvarsfullt förhållningssätt till digital teknik, för att

> kunna se möjligheter och förstå risker samt kunna värdera information. Utbildningen ska därigenom ge eleverna förutsättningar att utveckla *digital kompetens* […] (Skolverket, 2017b, s. 9).

Som ett led i detta uppdrag har användandet av digitala verktyg lagts till i kursplanerna för ett flertal ämnen. I teknik och matematik har förutom användande av digitala verktyg, även programmering lagts till i kursplanerna. I syftet i kursplanen för matematik kan det läsas att eleverna ska ”genom undervisningen ges möjligheter att utveckla kunskaper i att använda digitala verktyg och programmering för att kunna undersöka problemställningar och matematiska begrepp, göra beräkningar och för att presentera och tolka data” (Skolverket, 2017b, s. 56). Detta för att eleverna, som är vana användare av digital teknik, också ska ges möjlighet att utveckla förståelse för hur tekniken fungerar, vad den används i för olika yrken och områden, hur den påverkar samhället, samt utveckla ett kritiskt och ansvarsfullt förhållningssätt. Genom att få grundläggande kunskaper inom programmering kan eleverna förstå hur tekniken kan användas för att underlätta deras matematiska arbete, och få en grund för vidare lärande. Eleverna ges också bättre förutsättningar att kunna ta sig an framtidens teknik med gott resultat om de redan har en god förförståelse för hur digitala verktyg kan användas i matematiska sammanhang (Skolverket, 2017a). I kommentarmaterialet för kursplanen i matematik beskrivs det också hur digitala verktyg kan visualisera och konkretisera abstrakta fenomen, och är därmed ett hjälpmedel för lärande i matematik (Skolverket, 2017a).

För lågstadiets del har några få ändringar gjorts i det centrala innehållet för årskurs 1-3. Gällande programmering kan man under rubriken *Algebra* läsa att eleverna ska undervisas i ”hur entydiga stegvisa instruktioner kan konstrueras, beskrivas och följas som grund för programmering” och ”symbolers användning vid stegvisa instruktioner” (Skolverket, 2017b, s. 57). Tanken är då att eleverna ska ges grundläggande kunskaper i programmering, och utveckla ett förhållningssätt till detta. Kunskaperna ska sedan kunna användas i ämnesöverskridande arbete, exempelvis tillsammans med teknik eller samhällskunskap (Skolverket, 2017a). I kunskapskraven för årskurs 3 står dock inget om programmering. Däremot nämns det att eleverna ska kunna följa instruktioner, använda symboler samt föra och följa matematiska resonemang om mönster (Skolverket, 2017b).

2.3 Varför programmering i skolan?

Vi lever idag i ett informationssamhälle där vi är beroende av all teknik som finns omkring oss. För våra elever som är uppvuxna med tekniken är den ett naturligt inslag i vardagen, och de har stor vana utav att använda den. Deras generation kallas därför ibland för ”digitala infödingar”, ett smeknamn som egentligen är ganska missvisande (Mannila, 2017). Barnen visar snabbhet och säkerhet när de hanterar digitala verktyg, vilket ger ett intryck av stor digital kompetens. Men i själva verket har de sällan särskilt mycket kunskap om och hur tekniken fungerar, något som Karin Nygårds (2015) kallar ”digitalt ytflyt”. Barnen kan utnyttja tekniken till en viss del, då främst för nöjes skull, men de saknar kunskapen och förståelsen för det som finns under ytan. Likt en andraspråkselev ger sken utav att förstå mer svenska än hen egentligen gör, så ger den ”digitala infödingens” snabbhet och säkerhet sken av större digital kompetens än hen egentligen besitter (Mannila, 2017; Nygårds, 2015). Att ge eleverna förutsättningar för att utveckla digital kompetens är en del i skolans uppdrag (Skolverket, 2017b). Ett sätt att utveckla den digitala kompetensen är genom att lära sig programmering. Genom att besitta kunskaper i programmering utvecklas alltså en förståelse för hur den digitala tekniken fungerar, och ökar ens digitala kompetens (Kafai & Burke, 2013).

Redan under 1960-talet talade man om att barn skulle få lära sig att programmera, men det var först när datorn under 1980-talet blev allt vanligare i vardagen som programmering infördes i läroplaner runt om i världen. Datorn ansågs vara framtiden och för att elever skulle kunna använda sig utav datorer och utnyttja deras fulla kapacitet så krävdes en förståelse för hur datorer fungerar. Under slutet av 1990-talet blev datorerna mer lättanvända och det krävdes inte längre att man kunde programmera för att få sin dator att fungera. Varför skapa program och verktyg själv, när de redan finns färdiga att köpa? Fokus skiftade då från att förstå datorns uppbyggnad till att lära sig använda datorn, och programmering byttes ut mot IKT, informations- och kommunikationsteknik (Benton, Hoyles, Kalas, & Noss, 2017; Kafai & Burke, 2013; Mannila, 2017). Nu under 2010-talet har dock pendeln svängt tillbaka, och för att elever ska kunna utveckla den digitala kompetens som krävs i dagens informationssamhälle så anses det inte längre vara tillräckligt att bara kunna använda tekniken; man måste också förstå hur den fungerar, och därför får programmering återigen plats i läroplaner världen över (Kafai & Burke, 2013; Mannila, 2017).

2.4 Programmering och matematik

Att det finns en koppling mellan programmering och matematikämnet är inget nytt påfund. Matematikern och datavetaren Seymour Papert skrev i sin bok *Mindstorms – Children, Computers, and Powerful Ideas*, som utkom 1980, om programmering som ett verktyg i skolan för att lära och förstå matematik (Papert, 1980). Han menade att barn kan bli digitalt kompetenta och lära sig att behärska datorer och programmering, och eftersom programmering kräver att man använder matematiska strategier och sin problemlösningsförmåga, så kan datorer användas för att ge mening åt matematiken. Även elevernas matematiska språk utvecklas vid programmering och användande av datorer, eftersom man endast kan kommunicera med datorn med hjälp av det matematiska språket. Paperts tanke var alltså att programmering kan hjälpa fler elever att utveckla sina matematiska förmågor, eftersom det ger möjlighet att lära matematik på ett meningsfullt sätt genom skapande (Papert, 1980). En tanke som också forskare efter Papert är eniga om, eftersom programmering blir matematikundervisning på ett kreativt sätt som uppmuntrar till lek, och leken är ett naturligt sätt för barn att lära sig lösa problem (Gadanidis, Hughes, Minniti, & White, 2017).

Förutom att kunskaper i programmering utvecklar elevers problemlösningsförmåga, visar även senare forskning att förmågan till logiskt, systematiskt tänkande utvecklas, något som blir allt viktigare i vår samtid, inte minst inom matematik (Gadanidis m.fl., 2017; Helenius m.fl., 2017; Wing, 2006). Det systematiska, strukturella tänkandet där man löser ett problem steg för steg kallas också för datalogiskt tänkande (Helenius m.fl., 2017). Jeannette M. Wing, forskare inom datavetenskap, beskriver det datalogiska tänkandet som något alla borde sträva efter att lära sig, inte bara de som arbetar med datorer och programmering, eftersom det är en så pass nyttig förmåga att behärska för alla typer av problemlösning, även den vi möter i vardagen. Datalogiskt tänkande handlar alltså inte om att människan ska tänka som en dator, utan det är ett sätt att tänka för att lösa problem och skulle därför kunna ses som en grundläggande färdighet för att fungera i ett modernt samhälle (Wing, 2006).

Forskning visar också att programmering ger eleverna möjlighet att få använda sina matematiska kunskaper på ett meningsfullt sätt. Det eleverna gör i sin programmering syns i det färdiga resultatet, oavsett om det är ett spel de har programmerat eller en bana åt en robot, vilket belyser och underlättar förståelsen för kopplingen mellan den abstrakta matematiken och de konkreta upplevelserna. Den abstrakta matematiska processen görs alltså synlig i det programmerade resultatet. Gör man förändringar i den kod man skrivit, ändras också resultatet (Benton m.fl.,

2017; Gadanidis m.fl., 2017; Skolverket, 2017a). Använder man sig dessutom utav robotar i sin programmeringsundervisning blir dessa fysiska representationer utav de matematiska mönstren och processerna, vilket konkretiserar ytterligare (Gadanidis m.fl., 2017). Forskare beskriver också programmeringsundervisning som ett område med ”low floor – high ceiling”, det har alltså en låg lägstanivå och en hög högstanivå. Detta innebär att man kan göra undervisningen mycket enkel, och den kan därmed genomföras med gott resultat med de yngre eleverna som saknar förkunskaper om programmering. Den kan också göras mycket komplicerad och utmana elever med desto mer förkunskaper (Gadanidis m.fl., 2017; Hughes, Gadanidis, & Yiu, 2017). Undervisningen kan därför individanpassas efter elevernas olika nivåer och vara utmanande för alla.

Man kan då fråga sig varför programmering har saknats i kursplanen för matematik i grundskolans läroplan fram tills nu, när det är ett område med så många fördelar. Troligtvis för att det egentligen inte finns några studier som visar att eleverna blir bättre problemlösare i matematiska sammanhang genom att lära sig datalogiskt tänkande, eller att de får en bättre förståelse för det abstrakta. Anledningen till detta är att de kunskaper man får med sig i programmeringsundervisningen är svåra att transferera, det vill säga omvandla och utnyttja i andra sammanhang (Benton m.fl., 2017; Nygårds, 2015). Däremot finns det heller inget som säger att elevernas skolprestationer inte förbättras om de har utvecklat kunskaper inom programmering och datalogiskt tänkande. Om man som lärare är medveten om svårigheterna med att omvandla kunskaperna och aktivt arbetar med att överbrygga dem så borde ändå det datalogiska tänkandet kunna bli en god tillgång för elevernas framtida studier (Nygårds, 2015).

2.5 Programmering och algebra

I Lgr11 har programmering placerats under kunskapsområdet algebra i det centrala innehållet för matematik tillsammans med bland annat mönster i talföljder och geometriska mönster (2017b). Tanken bakom kunskapsområdet algebra är att eleverna ska inhämta en grundläggande algebraisk kunskap, vilket kan beskrivas som att man kan uttrycka beräkningar på ett generellt sätt, exempelvis genom att använda sig av bokstavsbeteckningar eller symboler. Att kunna föra generella resonemang kan sedan utnyttjas i andra matematiska sammanhang, såsom problemlösning och geometri (Skolverket, 2017a). Den algebraundervisning som sker på

lågstadiet kallas pre-algebra, och fungerar som en inkörsport till det fortsatta arbetet med algebra. Fokus ligger på att eleverna ska lära sig att resonera, fundera, generalisera, se samband och mönster och kunna sätta ord på vad de ser, för att utveckla deras algebraiska tänkande (Olteanu, 2014).

Att arbeta med mönster är en del som tar stor plats i lågstadiets algebraundervisning (Skolverket, 2017a). Det är ett brett område där eleverna får undersöka strukturer och logiska samband, men där de också tillåts vara kreativa i sitt skapande utav mönster. Mönstren kan representeras av vad som helst; konkreta föremål som exempelvis klossar, men också av siffror, geometriska figurer eller symboler. Eleverna lär sig också att generalisera genom att hitta gemensamma mönster i problem och lösningar; kunskaper som de sedan också har nytta av vid lösande av andra problem med andra representationer än de mönster de redan löst (Jahnke, 2011). Även inom programmering krävs det att eleverna analyserar det problem de har framför sig och finner likheter och mönster för att lösa problemet. Genom att se de logiska sambanden och genom kreativt tänkande kan de skapa en kod. Mönstren i koderna representeras av symboler (Helenius m.fl., 2017; Mannila, 2017).

3 Teoretiska utgångspunkter

I detta avsnitt redovisas de teorier som denna studie och dess analys har som utgångpunkter. Inledningsvis presenteras ett analysverktyg för appar, som kommer att användas som grund för mitt eget analysverktyg. Därefter redogörs teorier om tidig algebraundervisning, då programmering som tidigare nämnt står under algebra i grundskolans läroplan. Således bör teorier om algebra finnas med i analysen av apparna.

3.1 Analysverktyg för appar

Det har under årens lopp utarbetats flera olika verktyg och modeller för att analysera digitala lärresurser. Syftet med dessa är att underlätta för lärare i arbetet med att värdera och välja digitala lärresurser som passar lektioners syfte och mål (Palmér & Helenius, 2016). Eftersom det är appar som ska analyseras i denna studie, är det analysmodeller för just detta som bör användas.

En sådan modell är Walkers (2011) analysverktyg som baserats på sex kriterier som visats vara viktiga för lärare vid val av appar för undervisning: koppling till styrdokument, autenticitet, feedback, differentiering, användarvänlighet och motivation. I verktyget poängsätts apparna efter i vilken grad de uppfyller varje kriterium, ju högre poäng desto bättre. Priset på apparna upplevs också som viktigt av lärare, men eftersom det nödvändigtvis inte behöver påverka appens kvalitet är det inte ett kriterium.

Endast Walkers (2011) analysverktyg kan inte hjälpa mig att uppnå syftet med denna studie, eftersom det endast är behjälpligt vid en analys av apparnas lämplighet vid undervisning, men inte av deras matematiska innehåll. Alla kriterier är inte heller relevanta för denna studie. Analysverktyg kan dock kombineras eller manipuleras beroende på vilka kriterier man utgår ifrån i sin analys (Palmér & Helenius, 2016). Följande fyra kriterier från Walkers (2011) analysverktyg kommer därför att användas vid analysen i denna studie:

- **Koppling till styrdokument:** Då arbetet med appar i undervisningen ofta är ett självständigt arbete för eleverna krävs det att innehållet i apparna stämmer överens med läroplanen. Detta för att det ska bli en meningsfull aktivitet i elevernas lärande utan direkta instruktioner och genomgångar från läraren.

- **Feedback:** Det är viktigt hur apparna ger feedback till eleverna för att arbetet ska ses som effektivt och motiverande. Att återkopplingen är konstruktiv och kommer i rättan tid anses viktigt för att apparna ska hjälpa eleverna att förbättra sina prestationer. Feedbacken kan också komma som en sammanfattning av elevernas prestationer, och kan på så vis ge eleverna möjlighet att se sin utveckling över tid. Det är ett extra plus om den sammanfattade feedbacken också rapporteras till läraren, eftersom hen då kan anpassa fortsatt undervisning utifrån elevernas resultat.
- **Differentiering:** Lärarna i Walkers studie ansåg också att det var viktigt att appen går att nivåanpassa eller ställa in så den tränar en viss färdighet eller förmåga. Arbetet med appen blir då motiverande för alla elever, oavsett om de är låg- eller högpresterande, eftersom de ges chansen att lyckas och på så vis utvecklas även lärandet.
- **Användarvänlighet:** Användarvänligheten handlar om hur enkel appen är att använda, och i hur stor utsträckning eleven behöver stöd för att kunna arbeta med appen. De appar som anses vara bäst är de där eleven inte behöver särskilt mycket stöd utan kan använda och förstå appens innehåll på egen hand, och på så vis kan arbeta effektivt med appen under det självständiga arbetet (Walker, 2011).

3.2 Pre-algebra

Vad som egentligen innefattas i begreppet pre-algebra skiljer sig mellan forskare, då det inte är helt definierat. Pang (2016) beskriver dock i en redogörelse av forskning inom pre-algebra ett antal komponenter som är viktiga för den tidiga algebraundervisningen; mönster, användandet av symboler, och generaliseringar. Följande presenteras innebörden av dessa tre komponenter.

3.2.1 Mönster

Mönster finns överallt omkring oss; det finns i konstverk, på byggnader och i naturen och är därför något som tillhör vår vardag. Ordning och mönster är något människan har strävat efter i alla tider, något som syns redan hos små barn som ofta grupperar och storleksordnar sina leksaker. Kunskaper om mönster är alltså något som eleverna har med sig redan när de börjar skolan, men som behöver utvecklas för att kunna användas i matematiska sammanhang

(Bergsten, Häggström, & Lindberg, 1997). Detta eftersom förmågan att se mönster är flerdimensionell och dynamisk och inte kommer sig lika naturligt för alla, utan dess utveckling beror på bland annat på barnets kognitiva förmåga (Pang, 2016).

I alla mönster finns regler för hur de ser ut, och det är genom att lista ut reglerna som eleven vet hur mönstret ska fortsätta. Att få upptäcka regelbundenheten, strukturerna och de logiska sambanden är något eleverna förväntas möta i den tidiga algebraundervisningen (Jahnke, 2011). Genom exempelvis laborativt arbete kan eleverna lära sig att se matematiska mönster och på så vis genomskåda problemstrukturer och finna lösningar (Bergsten m.fl., 1997).

3.2.2 Användandet av symboler

Matematiken har ett eget uttryckssätt i form av symboler, vilket har ett stort utrymme inom algebran. Förstår man vad de olika symbolerna står för lär man sig också förstå hur räkneregler fungerar, och hur dessa kan användas (Bergsten m.fl., 1997). Symbolspråket är dessutom till stor hjälp vid generaliseringar och matematiska resonemang, vilket gör det till en viktig del av algebran (Pang, 2016). Genom att medvetet och regelbundet träna på att skriva, läsa, hantera och tolka symbolerna får eleven en förståelse för det matematiska symbolspråket. Förståelsen kan vara ytlig, det vill säga att eleven förstår hur man hanterar symbolen men inte vad den innebär. Man kan också ha en djupförståelse där eleven både vet hur symbolen hanteras och vad den står för, vilket är den typ av symbolkänsla som eftersträvas (Bergsten m.fl., 1997).

3.2.3 Generaliseringar

En av de mest grundläggande tankeformerna inom matematiken är att kunna göra generaliseringar. Att generalisera innebär att man kan utnyttja en inhämtad kunskap även i andra sammanhang, men också att man har den redan inhämtade kunskapen som grund och sedan kan rekonstruera eller expandera kunskapen (Bergsten m.fl., 1997). Genom generaliseringar är det möjligt att resonera sig fram till lösningar på matematiska problem, men också att studera problemen man står inför, förutsäga händelser i problemlösningsprocessen, samt bevisa och ge stöd för lösningarna man funnit. Generaliseringen bör dock ske medvetet för att den ska ses som en del av det algebraiska tänkandet (Pang, 2016).

4 Metodologisk ansats och val av metod

Syftet med denna studie är att ta reda på i hur stor utsträckning programmeringsappar som riktar sig till lågstadiet stämmer överens med vad som står i läroplanen, och hur väl dessa appar lämpar sig för att användas i matematikundervisning. Om syftet hade varit att ta reda på hur appar används i matematikundervisningen, hade observation varit en lämplig metod för datainsamling eftersom man då studerar människors beteende och därmed skulle kunna få en bild av hur användandet går till (Johansson & Svedner, 2010). Hade syftet varit att ta reda på vilka appar som används i undervisning och hur, hade en enkät eller intervju varit lämpliga metoder för att ta reda på hur lärare tänker och arbetar (Johansson & Svedner, 2010). Valet av metod beror alltså på vad man har för syfte (Åkerlund, 2017). I detta fall är syftet att ta reda på appars innehåll. Den metod som används för att undersöka studiens frågeställningar är därför en kvalitativ innehållsanalys.

4.1 Vald metod

Bryman (2011) beskriver innehållsanalys som en metod som främst förknippas med analys av texter och dokument, men att den är flexibel på så vis att den kan användas på många olika sorters ostrukturerad information. Den är objektiv och systematisk, vilket innebär att man i förväg klart och tydligt måste formulera regler för hur material ska kategoriseras, och att dessa regler sedan måste följas konsekvent så utrymmet för missvisande resultat blir så litet som möjligt. En följd av detta sätt att arbeta blir att alla som följer dessa regler bör uppnå ungefär samma resultat (Bryman, 2011). Viktiga aspekter att ha i åtanke vid genomförande av en innehållsanalys är att inte låta ens egna tolkningar ta för stor plats när man utformar analysschemat och att se till att man har en teoretisk utgångspunkt för sin analys (Bryman, 2011).

4.2 Källdokument

För att välja ut de appar som skulle ingå i studien skickades en förfrågan ut i två Facebookgrupper för lågstadielärare om vilka appar de använder sig utav i sin programmeringsundervisning. Eftersom undervisning i programmering ännu inte är obligatoriskt är det inte alla lärare som undervisar i ämnet. Genom en förfrågan i sociala medier fick jag möjlighet att nå ut till en större massa, och därmed troligtvis fler som undervisar i

programmering, än om jag endast hade tagit kontakt med lärare i mitt närområde. Anledningen till att jag ville ta reda på vilka appar som redan används av lärare istället för att själv söka upp appar på internet är dels för att göra en avgränsning i djungeln av appar, men också för att det, som Bryman (2011) också framhåller, är lätt hänt att man i mängden av information som finns på internet får sökresultat som är missvisande och oanvändbara. Det är dessutom mer meningsfullt och mer välbehövligt att analysera appar som faktiskt används än appar som ingen känner till. För att utöka mina källdokument ytterligare ställde jag samma fråga även till lärare på skolor i mitt närområde som jag vet undervisar i programmering. Jag beslutade mig också för att analysera de appar som Mannila (2017) och Nygårds (2015) tipsar om i sina böcker om undervisning i programmering.

Syftet med denna studie är att ta reda på hur väl programmeringsappar stämmer överens med styrdokumenten för matematik, med kriteriet att de ska rikta sig mot elever i lågstadieåldrarna, det vill säga 6-10 år. Andra kriterier är att apparna måste finnas tillgängliga i Apples appbutik App Store, eftersom studien genomförs på en iPad, samt att det inte krävs några fysiska verktyg såsom robotar eller klossar för att använda apparna då detta inte fanns att tillgå.

Materialinsamlingen gav 22 appar, och efter att uteslutningar gjorts efter ovan nämnda kriterier återstod totalt 14 appar. Apparnas uppbyggnad gör att de kan delas in i två kategorier. Den ena kategorin är spelappar, där apparna är uppbyggda som spel och användarens uppgift är att programmera en figur att lösa uppgifter eller ta sig från start till mål. Användaren får då ofta poäng eller liknande belöningar för sin insats. Den andra kategorin är vad jag valt att kalla skaparappar, där användaren får skapa egna spel, filmer eller liknande. Användaren kan sedan dela sina skapelser med andra. Nedan ges en beskrivning av apparna som ingår i studien.

Tabell 1. *Beskrivning av apparna i studien.*

Appens namn	**Typ av app**	**Beskrivning**
A.L.E.X	Spelapp	Roboten A.L.E.X. ska i olika banor programmeras så den steg för steg tar sig från start till mål. Användaren kan också skapa egna banor åt A.L.E.X.
Bee-Bot	Spelapp	Användaren ska programmera ett bi att steg för steg

		ta sig från start till mål. Användaren samlar poäng och det går på tid. Ju snabbare användaren anger rätt programmering, desto högre poäng.
Blue-Bot	Spelapp	En robot ska programmeras att ta sig steg för steg från start till mål. Användaren kan välja att följa en färdig programmering, eller skapa en egen.
Box Island	Spelapp	Användaren ska programmera en figur att steg för steg gå en bana och samla in tre stjärnor. Svårighetsgraden ställs in efter ålder (6-8, 9-11, 12+).
Coda Game	Skaparapp	Användaren får lära sig att programmera egna spel. Spelen kan sedan delas med andra, och man kan spela spel som gjorts av andra användare.
CodeSpark Academy: The Foos	Spelapp & skaparapp	Användaren ska programmera figurer att steg för steg lösa problem i olika banor. Det finns även minispel där olika färdigheter inom programmering kan tränas. Appen har också spelskaparläge där användaren får skapa egna spel, och spela spel som programmerats av andra användare. Läraren har möjlighet att ge eleverna individuella inloggningar.
Hopscotch	Skaparapp	Användaren får programmera egna spel med blockbaserad programmering.
Kodable	Spelapp	Användaren ska programmera en prick att rulla från start till mål genom att ge instruktioner om vilket håll pricken ska rulla. Läraren kan ge eleverna individuella inloggningar, och då nivåanpassa för varje elev.
LEGO: Fix the Factory	Spelapp	En robot ska programmeras att steg för steg ta sig igenom olika banor i en fabrik.
Lightbot: Code Hour	Spelapp	En robot ska programmeras att steg för steg ta sig igenom olika banor och tända lampor på golvet. För

		att klara banan krävs det att alla lampor tänds.
Pyonkee	Skaparapp	Användaren kan skapa korta filmer och berättelser med det blockbaserade programspråket Scratch.
Scratch Jr	Skaparapp	Användaren kan skapa korta filmer och berättelser med det blockbaserade programspråket Scratch. Programspråket i appen har förenklats genom att det är symboler på blocken istället för text, som det vanligtvis är i Scratch.
Swift Playgrounds	Spelapp & skaparapp	Användaren lär sig att programmera med det textbaserade programspråket Swift. Användaren får först lära sig grunderna i ett spel där en figur ska ta sig från start till mål eller utföra uppdrag, och kan sedan programmera egna spel i appen.
Tynker	Spelapp	Appen innehåller två spel: det ena handlar om en astronaut och det andra om en drake. I båda spelen ska användaren programmera figurerna att steg för steg ta sig igenom olika banor. Användaren kan välja mellan blockbaserad och textbaserad programmering.

4.3 Analysmodell

Analysmodellen har delats in i två delar: den första handlar om apparnas matematiska innehåll, och den andra om apparnas lämplighet i undervisning. Varje del har sedan delats in i olika analysområden. Analysområdena och deras kriterier i del ett har hämtats från vad som står om programmering och algebra i kursplanen för matematik åk 1-3 (Skolverket, 2017b), vad forskning anser vara viktiga aspekter i den tidiga algebraundervisningen, samt kriteriet koppling till styrdokument från Walkers (2011) analysverktyg för appar. I del två har samtliga analysområden och kriterier baserats på Walkers (2011) analysverktyg. På vänster sida i modellen anges analysområdena, och på höger sida de kriterier appen förväntas uppfylla. Ju fler kriterier som

uppnås, desto bättre bör en app lämpa sig för matematikundervisning på lågstadiet. Eftersom information på internet snabbt och ständigt förändras betonar Bryman (2011) vikten av att notera det datum man hämtat informationen, för att minska förvirring och frustration när någon annan försöker hitta det man refererar till. Den snabba utvecklingen gäller även för appar, därför anges förutom appens namn också den version av appen som analyserats. En närmare beskrivning av hur analysmodellen ska användas finns i Bilaga 1. Analys av samtliga 14 appar finns i Bilaga 2. Resultatet av analyserna presenteras vidare i studien i form av tabeller.

Appens namn (version) Kort beskrivning av appens innehåll	
Matematiskt innehåll	**Kriterier**
Mönster	Användaren får skapa egna mönster
	Användaren får jämföra mönster
	Användaren får resonera kring mönsters uppbyggnad
Symboler	Användaren får möta olika typer av symboler
	Användaren får förståelse för olika symbolers betydelse
	Användaren får skapa med symboler
Generaliseringar	Användaren måste medvetet utnyttja kunskaper som inhämtats tidigare i appen
Stegvisa instruktioner	Användaren får följa stegvisa instruktioner
	Användaren får skapa egna stegvisa instruktioner
Pedagogisk utformning	**Kriterier**
Användarvänlighet	Lättförstådda instruktioner
	Möjligt att välja språk
	Möjligt att pröva sig fram
Differentiering	Möjligt att nivåanpassa
	Möjligt att träna en viss färdighet
Feedback	Eleven får feedback om sitt eget resultat
	Läraren får feedback om elevens resultat

Figur 1. Studiens analysmodell.

4.4 Primäranalys

Inför studiens analys gjordes en primäranalys för att testa analysmodellen. Primäranalysen genomfördes på så vis att ett par appar provspelades för att testa om de kriterier som fanns i modellen var möjliga att analysera. I det centrala innehållet i matematik för årskurs 1-3 står det att förutom att skapa och följa stegvisa instruktioner ska eleverna även beskriva stegvisa

instruktioner (Skolverket, 2017b), vilket därför fanns med som ett kriterium i den ursprungliga analysmodellen under analysområdet *stegvisa instruktioner*. Detta visade sig dock vara svårt att analysera, då modellen utgår ifrån att eleverna ska använda apparna som självständigt arbete, och att beskriva stegvisa instruktioner är en aktivitet som är svår att bedriva självständigt. Detta kriterium plockades därför bort ur modellen.

Ett annat analysområde som fanns med i den ursprungliga modellen var *autenticitet*, som placerades under pedagogisk utformning. Walker (2011) beskriver i sitt analysverktyg autenticitet som kvaliteten på uppgifterna som eleverna möter i appen. Under primäranalysen beslutades det dock att autenticiteten delvis redan analyserades i det matematiska innehållet, och *autenticitet* byttes därför ut mot *feedback*. *Feedback* lades till då det är en viktig aspekt att analysera för att ta reda på appars lämplighet i undervisning, då det både ger elev och lärare en rapportering om resultat samt kan verka motiverande för eleverna (Walker, 2011).

Förutom revidering av analysmodellen användes också primäranalysen till att formulera instruktionerna till analysmodellen (se Bilaga 1). I primäranalysen samlades exempel från uppgifterna i apparna som användes som exempel i instruktionerna.

4.5 Genomförande

Efter att primäranalysen och revideringen av analysmodellen gjorts påbörjades studiens analys. Analysmodellen skrevs ut i pappersform, en modell till varje app. Modellen fanns på så vis tillgänglig medan appen spelades och analyserades, och möjligheten fanns därför att skriva ner kommentarer och funderingar om appen under analysens gång. Om ett kriterium från analysmodellen inte stöttes på i appen ströks det i modellen. Apparna analyserades så långt som det gavs åtkomst i deras gratisversioner. För vissa appar innebar det att hela appen analyserades, för andra endast några banor.

När analyser gjorts av samtliga 14 appar renskrevs analysmodellerna genom att fyllas i digitalt. Appens namn, version, samt en kort beskrivning angavs, och de kriterier som appen inte uppnått raderades och lämnades tomma (se Figur 2). När resultatet skulle sammanfattas gjordes tabeller för varje analysområde där varje kriterium fick en kolumn, och om en app uppnått kriteriet markerades det med ett kryss (se kapitel 5 Resultat och analys).

A.L.E.X. (version 1.64) Roboten A.L.E.X. ska i olika banor programmeras så den steg för steg tar sig från start till mål. Användaren kan också skapa egna banor åt A.L.E.X. Användaren får möta sekvensering.	
Matematiskt innehåll	**Kriterier**
Mönster	Användaren får skapa egna mönster
	Användaren får resonera kring mönsters uppbyggnad
Symboler	Användaren får möta olika typer av symboler
	Användaren får förståelse för olika symbolers betydelse
	Användaren får skapa med symboler
Generaliseringar	Användaren måste medvetet utnyttja kunskaper som inhämtats tidigare i appen
Stegvisa instruktioner	
	Användaren får skapa egna stegvisa instruktioner
Pedagogisk utformning	**Kriterier**
Användarvänlighet	Lättförstådda instruktioner
	Möjligt att pröva sig fram
Differentiering	
Feedback	Eleven får feedback om sitt eget resultat

Figur 2. Exempel på en ifylld analysmodell efter analys av en app.

4.6 Forskningsetiska principer

Studien genomfördes i enlighet med Vetenskapsrådets beskrivning av de forskningsetiska principerna för humanistisk-samhällsvetenskaplig forskning (Vetenskapsrådet, 2002). Detta innebär att studien uppfyller de fyra huvudkraven; informationskravet, samtyckeskravet, konfidentialitetskravet och nyttjandekravet (Vetenskapsrådet, 2002). Informationskravet och samtyckeskravet uppfylldes genom att deltagarna informerades om studiens syfte, och om att delgivandet av information om vilka appar de använder i sin undervisning var frivilligt. I enlighet med konfidentialitetskravet användes eller bevarades inga personuppgifter, eftersom det inte var relevant för studien. Vid materialinsamlingen var det endast informanternas svar som samlades in, inga namn. För att uppfylla det fjärde och sista kravet, nyttjandekravet, användes de insamlade uppgifterna endast som källdokument till analysen i denna studie.

4.7 Validitet, reliabilitet och generaliserbarhet

Validiteten handlar om studiens giltighet; har studien undersökt det som avsågs, och täcker resultatet hela området för studien (Johansson & Svedner, 2010). För att öka validiteten i denna studie har mycket arbete lagts ner på skapandet av analysmodellen. Eftersom det inte fanns några analysmodeller som i sin helhet räckte till för denna studie, fick en ny modell skapas för att besvara studiens syfte. Genom att se till att det finns en tydlig koppling mellan analysmodellen och studiens syfte och frågeställningar, eftersträvades en hög validitet. Validiteten kan också påverkas av hur människor tolkar begrepp på olika sätt (Johansson & Svedner, 2010). I inledningen och bakgrunden till studien har centrala begrepp definierats, så att de ska tolkas på samma sätt av alla som läser dem, och därmed stärks begreppsvaliditeten.

Reliabilitet står för tillförlitligheten och mätnoggrannheten i studien (Johansson & Svedner, 2010). Innehållsanalys är en metod som bör ha hög reliabilitet, eftersom analysen sker efter tydligt formulerade kategorier och regler, och därför borde alla som genomför analysen med samma verktyg få samma resultat. Detta kräver dock att analysmodellen beskrivs så väl att det även kan användas av andra (Johansson & Svedner, 2010). Detta har gjorts dels genom texten som förklarar modellen och beskrivningen av studiens genomförande, dels med hjälp av en instruktion till analysmodellen (Bilaga 1) där det ges förklaringar och exempel på hur analysmodellen bör användas och vad kriterierna står för. Objektet som analyserats i denna studie är elektroniska källor, vilka har en tendens till att ofta uppdateras eller till och med försvinna (Bryman, 2011). Detta kan minska reliabiliteten, eftersom det då kan bli svårt att göra om analysen på samma objekt. Den minskade reliabiliteten har motverkats genom att versionen av den analyserade appen har angetts, för att göra det möjligt att analysera samma objekt.

Generaliserbarhet har en nära koppling till reliabilitet och innebär att det resultat man får av analysen kan generaliseras till exempelvis alla appar inom kategorin utbildning inom programmering (Johansson & Svedner, 2010). Att göra generaliseringar utifrån resultatet i denna studie är svårt, eftersom apparna har olika upplägg och innehåll trots att de tillhör samma kategori. Generaliserberheten i det här fallet handlar istället om förekomst. Eftersom studiens källdokument valdes utifrån appar som redan används, eller rekommenderas att användas, i undervisning så kan resultatet generaliseras till att det är den här typen av appar och uppgifter som förekommer i programmeringsundervisningen på lågstadiet.

5 Resultat och analys

Resultatet av analysen delas in i två olika teman: matematiskt innehåll och pedagogisk utformning. Det första temat syftar till att besvara studiens första frågeställning om hur väl apparna stämmer överens med kursplanen i matematik och därmed lämpar sig för användning i matematikundervisningen på lågstadiet. Syftet med det andra temat är att analysera delar av applikationernas pedagogiska utformning för att ta reda på huruvida de lämpar sig för att användas i undervisning på lågstadiet överlag. Dessa två teman har sedan delats upp med de analysområden som återfinns i analysmodellen som underkategorier, där resultatet av varje analysområde presenteras sammanställt i tabeller. Respektive tema avslutas med en analys av resultatet med avstamp i studiens teoretiska utgångspunkter.

5.1 Matematiskt innehåll

5.1.1 Mönster

Tabell 2. *Förekomst av mönster i apparna.*

	A.L.E.X	Bee-Bot	BlueBot	Box Island	Coda Game	CodeSpark Academy: The Foos	Hopscotch	Kodable	LEGO: Fix the Factory	Lightbot: Code Hour	Pyonkee	Scratch Jr	Swift Playgrounds	Tynker	
Skapa egna mönster	X		X	X		X		X	X	X			X	X	**Tot: 9/14**
Jämföra mönster													X		**Tot: 1/14**
Resonera kring mönsters uppbyggnad	X		X	X		X		X	X	X			X	X	**Tot: 9/14**

Som redovisas i Tabell 2 är antalet appar där användaren får skapa egna mönster och resonera kring mönsters uppbyggnad samma. Dessa appar är uppbyggda som spel där användarens uppgift är att programmera en figur att ta sig från start till mål. För att göra detta måste användaren skapa ett mönster av symboler, och för att programmeringen ska stämma krävs det att användaren resonerar kring mönstrets uppbyggnad. Även appen *Bee-Bot* är ett sådant spel, där visas dock inte den programmering som användaren skapar och det blir därför inte något synligt mönster.

Appen *Swift Playgrounds* skiljer sig från de andra spel-apparna på så vis att programmeringen som skapas där är textbaserad, och mönstren skapas därför med textkommandon istället för symboler. Detta är också den enda appen som uppnår kriteriet ”jämföra mönster”. Detta eftersom användaren på en nivå får uppdraget att hitta buggar i en färdig programmering, och ges då möjligheten att jämföra den programmeringen med en egen för att lösa problemet.

Gemensamt för alla de appar som inte uppnår något kriterium, med undantag för tidigare nämnda *Bee-Bot*, är att de är appar där man som användare får skapa fritt; antingen egna spel eller korta filmer och berättelser.

5.1.2 Symboler

Tabell 3. *Förekomst av symboler i apparna.*

	A.L.E.X	Bee-Bot	BlueBot	Box Island	Coda Game	CodeSpark Academy: The Foos	Hopscotch	Kodable	LEGO: Fix the Factory	Lightbot: Code Hour	Pyonkee	Scratch Jr	Swift Playgrounds	Tynker	
Möta olika typer av symboler	X	X	X	X	X	X		X	X	X		X			**Tot: 10/14**
Förståelse för symbolers betydelse	X	X	X	X	X	X		X	X	X		X			**Tot: 10/14**
Skapa med symboler	X	X	X	X	X	X		X	X	X		X			**Tot: 10/14**

Samtliga tre kriterier för analysområdet ”symboler” uppnåddes av samma appar, där majoriteten var spelappar där användaren skulle programmera en figur att ta sig från start till mål. Programmeringen bestod då utav pilar som angav riktningen som figuren skulle förflytta sig, och andra symboler specifika för just det spelet. Detta med två appar som undantag: *Coda Game* och *Scratch Jr*. I *Coda Game* ska användaren programmera ett eget spel, där olika symboler ger olika funktioner i slutprodukten. *Scratch Jr* är en app med blockbaserad programmering där användaren får skapa korta filmer och berättelser. På varje block finns symboler som visar vad blocket gör i programmeringen. I samtliga tio appar som nådde kriterierna fick användaren möta olika typer av

symboler, och skapa med dessa för att nå appens mål. För att göra detta krävdes det en förståelse för symbolernas betydelse.

Hopscotch, Pyonkee och *Tynker* består liksom *Scratch Jr* av blockbaserad programmering. Dessa uppnår dock inte kriterierna, eftersom dessa har text på blocken istället för symboler. *Swift Playgrounds* är helt och hållet textbaserad.

5.1.3 Generaliseringar

Tabell 4. *Förekomst av generaliseringar i apparna.*

	A.L.E.X	Bee-Bot	BlueBot	Box Island	Coda Game	CodeSpark Academy: The Foos	Hopscotch	Kodable	LEGO: Fix the Factory	Lightbot: Code Hour	Pyonkee	Scratch Jr	Swift Playgrounds	Tynker	
Medvetet utnyttja kunskaper som inhämtats tidigare i appen	X	X	X	X		X	X	X	X	X	X	X	X	X	**Tot: 13/14**

Majoriteten av apparna blev stegvis svårare, och med jämna mellanrum introducerades nya moment som det krävdes att man som användare lärde sig för att sedan kunna utnyttja vidare i spelet. Detta gjorde att man fick reflektera över hur man gjort tidigare för att kunna lösa de nya problemen, med andra ord: generalisera. Endast en av de 14 analyserade apparna levde inte upp till detta kriterium: skaparappen *Coda Game* där användaren fick skapa egna spel. Oavsett antalet spel man skapar, ser ”arbetsbordet” där spelen skapas likadant ut, och genom att trycka på en symbol ges man en förklaring av dess innebörd. Vilket genererar till att man som användare inte medvetet måste reflektera över hur man gjort tidigare, utan kan trycka sig fram utan närmare eftertanke.

5.1.4 Stegvisa instruktioner

Tabell 5. *Förekomst av stegvisa instruktioner i apparna.*

	A.L.E.X	Bee-Bot	BlueBot	Box Island	Coda Game	CodeSpark Academy: The Foos	Hopscotch	Kodable	LEGO: Fix the Factory	Lightbot: Code Hour	Pyonkee	Scratch Jr	Swift Playgrounds	Tynker	
Följa stegvisa instruktioner			X												**Tot: 1/14**
Skapa stegvisa instruktioner	X	X	X	X		X	X	X	X	X	X	X	X	X	**Tot: 13/14**

Att skapa stegvisa instruktioner var, som visas i Tabell 5, något man som användare fick möta i samtliga appar utom en. Vare sig det handlade om att programmera en figur att ta sig från start till mål eller att skapa en kort film, så fick användaren skapa en programmering av stegvisa instruktioner för att få saker att förflytta sig. I appen som var undantaget, *Coda Game*, fick användaren inte möta några synliga stegvisa instruktioner. Där placerades en symbol som representerade ett kommando ut och gällde då under hela spelets gång utan att man som användare behövde specificera att det skulle vara så.

Kriteriet ”att följa stegvisa instruktioner” nåddes endast av en app: *Blue-Bot*. I denna app var det möjligt för användaren att välja om man själv ville skapa en stegvis instruktion åt roboten, eller om man ville följa en. Valde man att följa en stegvis instruktion, så dök en instruktion och en målflagga upp på spelplanen. Man skulle då som användare lista ut var roboten skulle starta för att ta sig i mål enligt instruktionen.

5.1.5 Analys av apparnas matematiska innehåll

Den första delen av analysmodellen behandlar apparnas matematiska innehåll med koppling till kursplanen i matematik i Lgr11 (Skolverket, 2017b). Den syftar på så vis till att ge svar på studiens första frågeställning om hur apparna stämmer överens med kursplanen i matematik. Då elevernas arbete med appar ofta är ett självständigt arbete utan genomgångar och instruktioner från läraren krävs det att innehållet i apparna stämmer väl överens med läroplanens skrivelser för att det ska vara en meningsfull aktivitet (Walker, 2011). Då programmering har placerats under

algebra i läroplanen, är det med utgångspunkt i kursplanen i matematik och teorier om pre-algebra som följande analys av resultatet görs.

Alla mönster har regler för hur de ska se ut, och det är genom att upptäcka regelbundenheter och logiska samband som mönstrets regler kan listas ut (Jahnke, 2011). I nio av de 14 analyserade apparna fick användaren skapa egna mönster och resonera över mönsters uppbyggnad för att kunna skriva sina programmeringar. För att programmeringen då skulle stämma, var användaren tvungen att ha förstått regeln för mönstret i programmeringen. Mönstret blir då ett sätt för användaren att upptäcka problemstrukturer och finna lösningar på problemen, så som det ofta är i matematik (Bergsten m.fl., 1997).

Tio av de analyserade apparna levde upp till samtliga tre kriterier inom analysområdet symboler. Förståelsen för symboler och deras användning kan enligt Bergsten, Häggström och Lindberg (1997) vara ytlig eller djup. Eleven har en ytlig förståelse för symboler när hen vet hur man hanterar en symbol, men inte förstår symbolens innebörd. Har eleven däremot en djupförståelse kan hen både hantera symbolen och förstå dess innebörd (Bergsten m.fl., 1997). I apparna får eleven möta olika typer av symboler, lära sig deras innebörd och på olika sätt använda dem, och apparna bör därför hjälpa eleven att utveckla en djupförståelse för symboler i största allmänhet. De appar som inte nådde kriterierna innehöll antingen blockbaserad programmering med text på blocken, eller var helt och hållet textbaserade och erbjöd därför inte användaren något möte eller arbete med symboler.

Förmågan att generalisera är viktig inom all matematik, men kanske framförallt inom algebran då det gör det möjligt att finna strategier och resonera sig fram till lösningar på matematiska problem (Bergsten m.fl., 1997; Pang, 2016). För att generaliseringen ska ses som en del av det algebraiska tänkandet är det dock viktigt att det är en medveten handling (Pang, 2016). I 13 av 14 appar fick användaren reflektera över och utnyttja kunskaper som hen inhämtat i tidigare nivåer för att kunna lösa de nya uppgifterna.

Att kunna skapa och följa stegvisa instruktioner beskrivs i kommentarmaterialet till kursplanen i matematik som grunden till att förstå programmering och hur programmering kan användas (Skolverket, 2017a). Att skapa stegvisa instruktioner får användaren möta i 13 av 14 analyserade appar, och kan därför ses som en färdighet elever kan lära sig med hjälp av sådana digitala lärresurser. Att följa stegvisa instruktioner fick användaren endast möta i en app, och en slutsats som kan dras av det är att det är en färdighet som är svårare att lära sig med hjälp av en app.

Skolverket (2017a) föreslår i kommentarmaterialet till kursplanen i matematik att detta istället kan vara en färdighet som tränas fysiskt genom att eleverna får skapa och följa varandras instruktioner.

5.2 Pedagogisk utformning

5.2.1 Användarvänlighet

Tabell 6. *Olika aspekter av användarvänlighet i apparna.*

	A.L.E.X	Bee-Bot	BlueBot	Box Island	CodaGame	CodeSpark Academy: The Foos	Hopscotch	Kodable	LEGO: Fix the Factory	Lightbot: Code Hour	Pyonkee	Scratch Jr	Swift Playgrounds	Tynker	
Lättförstådda instruktioner	X	X	X	X	X	X		X	X					X	**Tot: 9/14**
Möjligt att välja språk				X	X	X				X	X	X		X	**Tot: 7/14**
Möjligt att pröva sig fram	X	X	X	X	X	X	X	X	X	X	X	X	X	X	**Tot: 14/14**

I majoriteten av apparna gavs instruktioner som var enkla för användaren att förstå och följa. Instruktionerna bestod då ofta utav att det visades i appen hur man skulle göra vid varje nytt moment, eller att man blev introducerad steg för steg. I de appar som inte levde upp till kriteriet kunde instruktionerna till exempel vara endast på engelska, ha ett för högt tempo, bestå av mycket text, sakna förklaringar av nya begrepp, eller så fanns inga instruktioner alls.

I hälften av apparna fanns det möjlighet för användaren att välja språk. Som minst fanns det tre språk att välja mellan: svenska, engelska och spanska, och i andra appar fanns en mängd olika språk att välja. De appar där möjligheten att välja språk inte fanns var på engelska, med undantag för *Swift Playgrounds* där de nivåer där man lärde sig grunderna var på svenska och vidare nivåer var på engelska.

Möjligheten att pröva sig fram fanns i alla appar. I de appar som var uppbyggda som spel kunde användaren förlora poäng, stjärnor eller liknande om man gjorde fel, men det fanns inget högsta

antal försök och man hade som användare alltid möjlighet att spela om nivån för att förbättra sitt resultat. I de skapande apparna, där man skapade spel eller korta filmer, var det fritt att pröva sig fram hur mycket som helst.

5.2.2 Differentiering

Tabell 7. *Olika aspekter av differentiering i apparna.*

	A.L.E.X	Bee-Bot	BlueBot	Box Island	Coda Game	CodeSpark Academy: The Foos	Hopscotch	Kodable	LEGO: Fix the Factory	Lightbot: Code Hour	Pyonkee	Scratch Jr	Swift Playgrounds	Tynker	
Möjligt att nivåanpassa			X	X		X		X					X	X	**Tot: 6/14**
Möjligt att träna en viss färdighet			X			X		X		X		X	X	X	**Tot: 7/14**

Möjligheten att nivåanpassa var begränsad bland apparna. I de appar där det var möjligt kunde man som användare växla mellan nivåer eller åldrar (6-8, 9-11, 12+) för att göra det enklare eller svårare. I appen *Kodable* var det dessutom möjligt för läraren att via en lärarinloggning på internet ge varje elev en personlig inloggning till appen, och där också tilldela eleverna olika nivåer.

Att välja att träna en viss färdighet var möjligt i hälften av de analyserade apparna. I dessa appar var det tydligt utskrivet vad som tränades i de olika nivåerna, och det var då möjligt att gå tillbaka och öva extra vid behov. I appen *CodeSpark Academy: The Foos* fanns förutom ”huvudnivåerna” minispel där användaren gavs möjlighet att träna vissa färdigheter.

5.2.3 Feedback

Tabell 8. *Olika aspekter av feedback i apparna.*

	A.L.E.X	Bee-Bot	BlueBot	Box Island	Coda Game	CodeSpark Academy: The Foos	Hopscotch	Kodable	LEGO: Fix the Factory	Lightbot: Code Hour	Pyonkee	Scratch Jr	Swift Playgrounds	Tynker	
Eleven får feedback om sitt eget resultat	X	X	X	X		X		X	X	X			X	X	**Tot: 10/14**
Läraren får feedback om elevens resultat						X		X							**Tot: 2/14**

I samtliga appar som var uppbyggda som spel fick användaren feedback om sitt resultat i form av exempelvis poäng, stjärnor eller pengar. Ju fler poäng, stjärnor eller pengar, desto bättre resultat. I de skapande apparna fick användaren ingen rapport om sitt resultat.

I två av apparna var det möjligt för läraren att se elevernas resultat utan att behöva titta på varje elevs surfplatta. I *Kodable* och *CodeSpark Academy: The Foos* kan läraren skapa en lärarinloggning och där tilldela varje elev en personlig inloggning till skolläge. När eleverna sedan spelar i skolläge registreras deras resultat på lärarens sida, och läraren kan då följa varje elevs resultat därifrån.

5.2.4 Analys av apparnas pedagogiska utformning

Analysmodellens andra del behandlade apparnas pedagogiska utformning, och förväntades besvara studiens andra frågeställning om hur apparna lämpar sig för undervisning på lågstadiet. Resultatanalysen av den pedagogiska utformningen sker med teorier från Walkers (2011) analysverktyg som utgångspunkt.

Inför skapandet av analysverktyget, genomförde Walker (2011) en undersökning bland lärare för att ta reda på vad de ansåg vara viktiga aspekter vid val av appar till undervisning. Resultatet visade då att många lärare låter sina elever arbeta med appar individuellt, och att en bra app är en app där eleverna kan arbeta effektivt utan att behöva be om hjälp (Walker, 2011). Då är apparnas användarvänlighet, och instruktioner, avgörande. Majoriteten av apparna som ingick i denna studie hade instruktioner som var enkla att förstå. Dock hade de appar där användaren tilläts skapa fritt desto längre och mer komplicerade instruktioner. I endast hälften av apparna var det möjligt för användaren att ändra språk, vilket gör att de appar där det inte var möjligt kan

upplevas som svåra att förstå. Dock gav majoriteten av dessa appar sina instruktioner visuellt istället för att behöva läsas eller höras, vilket ändå höjde deras användarvänlighet eftersom det gjorde dem enklare att förstå. Gemensamt för alla analyserade appar var att de gav användaren möjlighet att pröva sig fram, vilket höjde användarvänligheten avsevärt. Även om eleven inte förstår instruktionerna så behöver hen inte sitta sysslolös och vänta på hjälp, utan kan då försöka pröva sig fram till förståelse under väntans gång och arbetet kan fortsatt ses som effektivt (Walker, 2011).

Ytterligare en viktig aspekt som Walker (2011) fann i sin undersökning var differentiering; att kunna nivåanpassa eller välja ut särskilda färdigheter för elever att träna. Han, och lärarna i hans studie, menar att individanpassning är nyckeln till både framgång och motivation hos eleverna och att det därför är något som bör erbjudas i en bra app (Walker, 2011). Möjligheten till differentiering fanns endast i ungefär hälften av de analyserade apparna. Många av apparna var utformade som spel, där man genom att klara en nivå låste upp nästa. Detta gjorde det möjligt för en elev som inte greppat ett visst moment att gå tillbaka och öva extra på det, men för en elev som är stark inom programmering fanns ingen möjlighet att hoppa över nivåer som var för enkla för att istället utmanas med svårare nivåer. Appen *Kodable* gav dock läraren möjligheten att helt och hållet styra och individanpassa svårigheten och utmaningarna för eleverna via en lärarinloggning. Detta gör appen, enligt Walker (2011), mer användbar för fler elever.

Slutligen analyserades också hur feedback gavs i apparna. Resultatet visade att de flesta av apparna gav eleverna information om hur de ligger till i form av belöningar efter avslutad nivå. Feedback såsom denna är viktig för att eleven ska kunna förbättra sina prestationer, men vad som är avgörande för att feedbacken ska ge effekt är att den är konstruktiv och kommer vid rätt tidpunkt och att feedbacken är effektiv på så vis att den hjälper eleven att hitta vägen mot det rätta svaret (Walker, 2011). Då apparna ständigt visar för eleven om något blivit fel (programmeringen stämmer inte), och eleven ges möjligheter att göra om och pröva sig fram så kan den feedback som ges av apparna med Walkers (2011) mått mätt ses som effektiv. Dock var det ytterst få, närmre bestämt bara två, appar som rapporterade elevernas resultat direkt till läraren. Walker (2011) beskriver sådan information som värdefull, eftersom den kan hjälpa lärare att göra val för vidare undervisning baserat på elevernas utveckling.

6 Diskussion

Följande avsnitt inleds med en metoddiskussion där det reflekteras och diskuteras kring valet av metod och studiens analysmodell. Därefter diskuteras studiens resultat i förhållande till forskning och litteratur, och hur resultatet svarar till studiens syfte och frågeställningar. Avslutningsvis redogörs några reflektioner som väcktes under studiens gång.

6.1 Metoddiskussion

För att besvara studiens syfte och frågeställningar om hur programmeringsappar stämmer överens med kursplanen i matematik i Lgr11 och lämpar sig för lågstadiets matematikundervisning krävdes en analys av apparnas innehåll, och valet av metod föll därför på en kvalitativ innehållsanalys. Analysen ger därmed svar på om ett analysområde förekommer i appen, men inte hur frekvent det förekommer då det hade krävt en kvantitativ analys. En kvantitativ analys kvantifierar, det vill säga fastställer mängden, data som analyseras (Bryman, 2011), och skulle därför kunna ge svar på hur ofta något förekommer. Frågan om eleverna får möta ett visst matematiskt innehåll besvaras i studien, men analysen ger inte svar på om det endast är en gång eller om det är vanligt förekommande i appen. Detta skulle förslagsvis kunna besvaras genom en vidareutveckling av analysmodellen där frekvens också beaktas.

Inför skapandet av analysmodellen studerades flera andra analysmodeller, som tillsammans med forskning och teoretiska utgångspunkter låg till grund för denna studies analysmodell. Efter primäranalysen och synpunkter från andra reviderades modellen och dess instruktioner ett flertal gånger innan den slutgiltiga analysen ägde rum. Detta till trots finns utrymme för vidareutveckling och förbättringar av modellen. Exempelvis fanns flera olika aspekter av språk i apparna, både talat, skrivet och visuellt, vilket gjorde det svårt att definiera vad som egentligen menades med kriteriet ”möjligt att välja språk” i analysområdet *användarvänlighet*, och därmed svårt att analysera. Definitionen som gjordes var att det gällde talat och skrivet språk, och att det syftade till att det skulle bli enklare för eleven att förstå appens instruktioner om hen kunde välja ett språk som hen förstod. Vad som dock upptäcktes under analysen var att det talade eller skrivna språket inte nödvändigtvis behövde vara avgörande för om eleven förstod appens instruktioner eller inte, då instruktionerna istället kunde ges visuellt genom att appen visade vad som skulle göras. Detta skapade funderingar kring om detta kriterium behövs överhuvudtaget för

att appen ska ses som användarvänlig, vilket man får reflektera över om modellen ska användas igen vid ytterligare analyser.

Ett annat kriterium som upplevdes som överflödigt efter analysen var ”möjligt att pröva sig fram”, även det under analysområdet *användarvänlighet*. Detta kriterium uppfylldes av samtliga appar i studien, och en slutsats som kan dras utifrån det är att det kan generaliseras till att det uppfylls av alla programmeringsappar som riktar sig till lågstadieåldrarna och därför inte är nödvändigt att ha med i analysen.

Gällande källdokumenten så togs beslutet att endast analysera den del apparna som hade gratis åtkomst. Detta beslut togs eftersom jag själv finansierade studien, och slutsumman skulle bli alldeles för hög om full åtkomst inhandlades till samtliga appar. Hur stor del av apparna som analyserade varierade därför, då vissa appar gav full åtkomst i gratisversionen, medan andra endast erbjöd att pröva en del av innehållet. Detta påverkar studiens resultat på så vis att en app kan ha uppgetts inte leva upp till något kriterium, när det i själva verket är så att man som användare inte får möta det i gratisversionen och att det skulle kunna dyka upp om man har full åtkomst. Det är därför viktigt att ha i åtanke att det är gratisversionerna som har analyserats, inte hela apparna.

6.2 Resultatdiskussion

6.2.1 Matematiskt innehåll

För att besvara studiens första frågeställning om hur väl programmeringsappar stämmer överens med kursplanen i matematik analyserades apparnas matematiska innehåll. Eftersom programmering placerats under algebra i läroplanen, analyserades innehållet utifrån teorier om pre-algebra. Resultatet av studien visar att flertalet av de appar som riktar sig till programmeringsundervisning på lågstadiet behandlar stora delar av de områden som Pang (2016) framhåller som viktiga inom pre-algebra; mönster, symboler och generaliseringar. Vad som dock kan diskuteras är om de används på samma vis inom programmering som inom algebra. Den aspekt där programmering och algebra stämmer bäst överens är mönster. Att arbeta med mönster handlar om att upptäcka regelbundenheter och logiska samband för att finna mönstrens regler (Jahnke, 2011). Detta krävs av eleverna för att de ska kunna skriva (textbaserad) eller sätta ihop (blockbaserad) de rätta programmeringarna i apparna.

När det gäller användandet av symboler så får eleverna möta flertalet olika symboler i apparna. Detta är dock symboler på en generell nivå, exempelvis flaggor och pilar, snarare än matematiska symboler som likhetstecken och bokstavssymboler som används inom algebra. Användandet av symboler inom algebra som det beskrivs i forskning ska underlätta för eleverna att förstå räkneregler och uttrycka matematiska resonemang (Bergsten m.fl., 1997; Pang, 2016). Att förstå vad en flagga innebär underlättar i programmeringssammanhang, men är det nödvändigt i ett matematiskt sammanhang?

Generaliseringar anses vara en grundsten i matematiken, och då kanske framförallt i algebran (Pang, 2016). Resultatet av studien visar att eleverna får göra generaliseringar i arbetet med apparna, men frågan är om den typ av generaliseringar som görs där stämmer överens med algebrans definition av generaliseringar. De generaliseringar som görs i apparna är i form av en progression, där eleven får utnyttja kunskaper som inhämtats i en enklare nivå för att klara en svårare nivå. Utmaningarna i appen blir svårare, men sammanhanget är detsamma. Generaliseringar ur algebrans synsätt handlar däremot om att kunna använda en inhämtad kunskap i andra sammanhang, och om att kunna rekonstruera den för att resonera sig fram till lösningar (Bergsten m.fl., 1997; Pang, 2016). Då utformning och upplägg i apparna skiljer sig åt kan det i vissa fall vara svårt att ta med sig kunskaper från en app och utnyttja i en annan, det vill säga i ett annat sammanhang.

Kopplingen mellan programmering och algebra upplevs stundom som rimlig, stundom som krystad, då arbetet med mönster, symboler och generaliseringar finns där men definitionen inte helt är densamma inom de båda områdena. I sökandet av forskning och litteratur inför denna studie var resultaten få där programmering och algebra kopplades samman, vilket tillsammans med resultatet av denna studie har väckt frågor om varför programmering egentligen placerats under algebra i läroplanen?

Det centrala innehållet i matematik för årskurs 1-3 säger också att eleverna ska få skapa och följa stegvisa instruktioner i undervisningen; ”hur entydiga stegvisa instruktioner kan konstrueras, beskrivas och följas som grund för programmering” (Skolverket, 2017b, s. 57). Resultatet av analysen visade att eleverna fick skapa stegvisa instruktioner i majoriteten utav apparna, men att de erbjöds att följa stegvisa instruktioner i ytterst få appar. Skolverket rekommenderar i kommentarmaterialet för matematik att eleverna ska få träna programmering fysiskt för att möta olika typer av stegvisa instruktioner på olika vis (Skolverket, 2017a).

6.2.2 Pedagogisk utformning

Analysen av apparnas pedagogiska utformning syftade till att besvara studiens andra frågeställning om hur väl apparna lämpar sig för undervisning på lågstadiet. Detta eftersom Palmér och Helenius (2016) skriver att det inte finns några regleringar för vem som får skapa en app, eller några kontroller av om appar som säljs som undervisningsappar verkligen lämpar sig för undervisning. Det är istället upp till lärarna själva att analysera huruvida apparna bör användas i undervisning eller ej. Majoriteten av apparna på marknaden är utvecklade utomlands (Nygårds, 2015). En hypotes väcktes därför i det inledande arbetet med studien om att en app kan stämma överens med läroplanen, men vara svåranvänd för eleverna på grund av exempelvis språkförbristningar, och skulle därför ändå inte vara lämplig att använda i undervisning.

Resultatet av analysen av apparnas pedagogiska utformning skiljde sig en hel del mellan apparna, och det var endast en app som levde upp till alla de kriterier som fanns i analysmodellen, *CodeSpark Academy: The Foos.* Av resultatet i övrigt framkom att de appar som nådde flest kriterier och därmed bör lämpa sig bäst att användas i undervisning på lågstadiet är de appar som är uppbygga som spel. Detta därför att instruktionerna i dessa appar var lättare att förstå, då de ofta visades visuellt. I skaparapparna var instruktionerna desto längre, mer komplicerade och gavs ibland bara på ett språk vilket gjorde dem desto svårare att använda för en lågstadieelev som ännu lär sig grunderna i programmering. Detta gjorde spelapparna mer användarvänliga; eleverna kan arbeta självständigt med apparna utan att behöva hjälp från lärare för att förstå appens instruktioner och innehåll (Walker, 2011).

I spelapparna fanns dessutom en progression som utmanar eleverna och som ger dem möjlighet att utveckla nya färdigheter inom programmering. Skaparapparna har ingen egen progression, utan kräver att eleven själv får pröva sig fram vilket kan innebära att inhämtandet av kunskap kan bli väldigt olika beroende på elevens intresse för programmering och tålamod att pröva sig fram. Spelapparna innehåller därmed en högre grad av differentiering, vilket innebär att svårigheten läggs på en nivå som passar den enskilde eleven (Walker, 2011). En lågpresterande elev kan stanna kvar på den nivå som passar hen, medan den högpresterande eleven kan arbeta vidare med mer utmanande nivåer. Ett alternativ för arbetet med skaparapparna är att läraren planerar progressionen och har gemensamma genomgångar med eleverna. Detta innebär dock att skaparapparna inte lämpar sig lika bra för individuellt arbete som spelapparna.

Spelapparna ger dessutom eleven direkt feedback i högre grad än skaparapparna i form av belöningar under arbetets gång. Det blir en formativ feedback som stöttar eleven i hens lärande och motiverar till bättre prestationer. Det ger dessutom både lärare och elev information för att se elevens styrkor och utvecklingsområden som kan vara till stöd för elevens fortsatta utveckling (Walker, 2011). Har eleven exempelvis fått alla stjärnor i nivåer som behandlar sekvensering, men endast en stjärna i nivåer som handlar om villkorssatser blir det tydligt att villkorssatser är ett utvecklingsområde för eleven. Skaparapparnas feedback ges när eleven väljer att testa den programmering hen har gjort och då ser om allt har blivit rätt. Risken finns då att feedbacken kommer i samband med resultatet istället för under arbetet med programmeringen, vilket gör feedbacken mer summativ än formativ. Arbetet med skaparapparna kräver också att eleven redan har en del programmeringskunskaper, dels för att kunna skriva eller sätta ihop programmeringen och dels för att kunna felsöka om programmeringen inte stämmer.

6.3 Slutsatser

Slutsatsen som dras utifrån resultatet av apparnas matematiska innehåll är att de på olika vis och i varierad utsträckning stämmer överens med kursplanen i matematik, men att det är de appar som är utformade som spel som stämmer bäst överens. Dessa appar skulle därför kunna lämpa sig för att användas i matematikundervisning, men då i kombination med andra lärresurser eller övningar för att helt och hållet täcka kursplanens skrivelser vad gäller programmering och tidig algebraundervisning. Resultatet visar att spelapparna också har en pedagogisk utformning som gör att det är dessa appar som lämpar sig bäst att använda i undervisning på lågstadiet överlag. Skaparapparna däremot skulle kunna lämpa sig bättre för äldre elever som redan satt grunderna till programmering, eller i undervisning av ett annat ämne än matematik.

6.4 Avslutande reflektioner

Tanken bakom denna studie var att genom en analys av programmeringsappars matematiska innehåll och pedagogiska utformning ta reda på vilka appar som lämpar sig för matematikundervisning på lågstadiet, och på så vis underlätta för lärare när det är dags att ta sig an utmaningen att undervisa i programmering. Slutsatserna som dras av studiens resultat är en

hjälpande hand i valet av appar, men det finns fortfarande fler saker att ta hänsyn till när man väljer appar att använda i sin undervisning.

Den 25 maj 2018 träder den nya dataskyddsförordningen, GDPR (General Data Protection Regulation), i kraft och gäller för hela EU. GDPR ersätter den tidigare personuppgiftslagen (PUL), och innefattar tydligare regler för hur personuppgifter får behandlas (Datainspektionen, 2018). I fackförbundet Lärarförbundets medlemstidning skriver Holmström (2018) att detta kan bli ett bekymmer vid val av appar till undervisning, då appar som utvecklats utanför EU också lagrar personuppgifter utanför EU, vilket innebär att de bryter mot GDPR-kraven. Detta blir ett extra stort bekymmer vid val av appar för programmeringsundervisning, då majoriteten av dessa appar utvecklats utanför EU (Holmström, 2018). Hur ska man som lärare förhålla sig till detta, då styrdokumenten säger att eleverna ska undervisas i programmering, men få lärresurser som lever upp till GDPR-kraven finns att tillgå?

Dessutom kvarstår frågan om kopplingen mellan programmering och algebra. Forskning och litteratur associerar i hög grad programmering med problemlösning (Gadanidis m.fl., 2017; Helenius m.fl., 2017; Papert, 1980; Wing, 2006), men få kopplar samman programmering och algebra. Här behövs mer forskning för att ytterligare belysa sambandet mellan dessa områden, och räta ut frågetecknet kring varför programmering placerats under algebra i *Läroplanen för grundskolan, förskoleklassen och fritidshemmet* (Skolverket, 2017b). Finns det någon vetenskaplig grund till varför programmering placerats där istället för under problemlösning?

Referenser

Benton, L., Hoyles, C., Kalas, I., & Noss, R. (2017). Bridging primary programming and mathematics: Some findings of design research in England. *Digital Experiences in Mathematics Education*, *3*(2), 115–138. https://doi.org/10.1007/s40751-017-0028-x

Bergsten, C., Häggström, J., & Lindberg, L. (1997). *Nämnaren TEMA: Algebra för alla*. Göteborg: Göteborgs Universitet.

Bryman, A. (2011). *Samhällsvetenskapliga metoder* (2:a uppl.). Stockholm: Liber.

Datainspektionen. (2018). Dataskyddsförordningen. Hämtad 02 maj 2018, från https://www.datainspektionen.se/dataskyddsreformen/dataskyddsforordningen/

Digitaliseringskommissionen. (2015). *Gör Sverige i framtiden - Digital kompetens* (SOU 2015:28). Stockholm: Digitaliseringskommissionen.

Gadanidis, G., Hughes, J. M., Minniti, L., & White, B. J. G. (2017). Computational thinking, grade 1 students and the binomial theorem. *Digital Experiences in Mathematics Education*, *3*(2), 77–96. https://doi.org/10.1007/s40751-016-0019-3

Helenius, O., Misfeldt, M., Rolandsson, L., & Ryan, U. (2017). Om programmering i matematikundervisning. Hämtad från https://larportalen.skolverket.se/#/modul/1-matematik/Grundskola/417_matematikundervisningmeddigitalaverktygII_%C3%A5k1-3/del_01/

Holmström, L. (2018). Vanliga appar och program kan stoppas. *Lärarnas tidning*, *18*(6), 9.

Hughes, J., Gadanidis, G., & Yiu, C. (2017). Digital making in elementary mathematics education. *Digital Experiences in Mathematics Education*, *3*(2), 139–153. https://doi.org/10.1007/s40751-016-0020-x

Jahnke, A. (2011). Det handlar om mönster. *Nämnaren*, (2). Hämtad från http://nbas.ncm.gu.se/namnaren

Johansson, B., & Svedner, P. O. (2010). *Examensarbetet i lärarutbildningen* (5:e uppl.). Uppsala: Kunskapsförlaget AB.

Jönsson, A. (2017). Formativ bedömning. I A. Hult & A. Olofsson (Red.), *Utvärdering och bedömning i skolan - För vem och varför?* (2:a uppl.). Stockholm: Natur & Kultur.

Kafai, Y. B., & Burke, Q. (2013). Computer programming goes back to school. *Phi Delta Kappan*, *95*(1), 61–65. https://doi.org/10.1177/003172171309500111

Mannila, L. (2017). *Att undervisa i programmering i skolan - Vad, hur och varför*. Lund: Studentlitteratur.

Mishra, P., & Koehler, M. J. (2006). Technological pedagogical content knowledge: A framework for teacher knowledge. *Teachers College Record*, *108*(6). Hämtad från http://www.tcrecord.org/

Nationalencyklopedin. (2018). Applikation. Hämtad från https://www-ne-se.bibproxy.kau.se/uppslagsverk/encyklopedi/l%C3%A5ng/applikation

Nygårds, K. (2015). *Koden till digital kompetens*. Stockholm: Natur & Kultur.

Olteanu, C. (2014). Vad är algebra och varför behövs den? Hämtad från https://larportalen.skolverket.se/#/modul/1-matematik/Grundskola/411_algebra%20%C3%A5k1-3/1_reflektionsomlarprocess/

Palmér, H., & Helenius, O. (2016). Analys av digitala programvaror. Hämtad från https://larportalen.skolverket.se/#/modul/1-matematik/Grundskola/416_matematikundervisningmeddigitalaverktyg_%C3%A5k1-3/5_analysavdigitalaprogramvaror/

Pang, J. (2016). A review of research that foregrounds the early algebra learner. Presenterad vid 13th International Congress on Mathematical Education, Hamburg.

Papert, S. (1980). *Mindstorms - Children, computers, and powerful ideas*. New York: Basic Books.

Skolinspektionen. (2011). *Litteraturöversikt för IT-användning i undervisningen*. Stockholm: Skolinspektionen.

Skolverket. (2016a). Digitala lärresurser. Hämtad från https://www.skolverket.se/skolutveckling/resurser-for-larande/itiskolan/digitala-larresurser

Skolverket. (2016b). *It-användning och it-kompetens i skolan - Skolverkets it-uppföljning 2015*. Stockholm: Skolverket.

Skolverket. (2016c). *Uppdrag om nationella it-strategier för skolväsendet*. Stockholm: Skolverket.

Skolverket. (2017a). *Kommentarmaterial till kursplanen i matematik - Reviderad 2017*. Stockholm: Skolverket.

Skolverket. (2017b). *Läroplan för grundskolan, förskoleklassen och fritidshemmet - Reviderad 2017*. Stockholm: Skolverket.

Utbildningsdepartementet. (2017). *Nationell digitaliseringsstrategi för skolväsendet*. Stockholm: Utbildningsdepartementet.

Vetenskapsrådet. (2002). *Forskningsetiska principer inom humanistisk-samhällsvetenskaplig forskning*. Stockholm: Vetenskapsrådet. Hämtad från http://www.codex.vr.se/texts/HSFR.pdf

Walker, H. (2011). Evaluating the effectiveness of apps for mobile devices. *Journal of Special Education Technology*, *26*(4), 59–63. https://doi.org/10.1177/016264341102600405

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33–35. https://doi.org/10.1145/1118178.1118215

Åkerlund, D. (2017). *Guide till akademiskt skrivande - Om att skriva rapporter, uppsatser och självständiga skriftliga arbeten på universitet och högskolor* (Upplaga 2.0). Karlstad: Karlstads universitet.

Bilaga 1

Instruktioner till analysmodellen

Analysmodellen är uppdelad i två delar: en som behandlar det matematiska innehållet i apparna, och en pedagogisk del bestående av aspekter som lärare anser vara viktiga vid val av app. Syftet med analysmodellens första del är att avgöra hur väl apparna stämmer överens med kursplanen i matematik i *Läroplan för grundskolan, förskoleklassen och fritidshemmet* (2017b), och därmed lämpar sig för att användas i matematikundervisningen. Syftet med den andra delen är att analysera delar av apparnas pedagogiska utformning för att ta reda på huruvida de lämpar sig för att användas i undervisning på lågstadiet överlag. På vänster sida i analysmodellen presenteras de olika analysområdena, och på höger sida står de kriterier som appen förväntas uppfylla.

Matematiskt innehåll

Den första delen i analysmodellen handlar om apparnas matematiska innehåll. Här analyseras hur apparna stämmer överens med vad som står om programmering i kursplanen för matematik i grundskolans läroplan, men också vad som står om mönster och symboler. Då programmering placerats under algebra i det centrala innehållet för matematik, har analysområdena och kriterierna valts ut med forskning om pre-algebra och läroplanens skrivelser som grund.

Mönster

Användaren får skapa egna mönster: Får användaren skapa mönster i appen? Mönstren kan bestå av symboler, siffror, bokstäver eller ord. Exempel: ↑→↑→, ”gå framåt” ”vänd höger” ”gå framåt” ”vänd höger”, ”moveForward( )” ”turnRight( )” ”moveForward( )” ”turnRight( )”.

Användaren får jämföra mönster: Ger appen användaren möjlighet att jämföra två eller fler mönster med varandra? Exempel: användaren kan vid felsökning av ett mönster (hitta buggar) jämföra det felaktiga mönstret med ett annat/eget mönster.

Användaren får resonera kring mönsters uppbyggnad: Får användaren resonera kring hur färdiga mönster eller egna mönster är uppbyggda? Exempel: fundera över hur mönstret ska skapas för att

programmeringen ska stämma, hitta fel (buggar) i mönstren, fundera över var i mönstret det har blivit fel.

Symboler

Användaren får möta olika typer av symboler: Får användaren möta olika typer av symboler i appen? Exempel: pilar, flaggor, lampor, boxar.

Användaren får förståelse för olika symbolers betydelse: Ges användaren möjlighet att skaffa förståelse för vad symbolerna betyder? Exempel: ↑ betyder ”gå framåt”, → betyder ”vänd höger”,  betyder ”start”,  betyder ”tänd lampa”,  betyder ”loop”/”slinga”.

Användaren får skapa med symboler: Får användaren använda symboler för att skapa? Här handlar det inte bara om att förstå symbolers betydelse, utan också hur de kan användas. Exempel: skapa mönster, skapa spel.

Generaliseringar

Användaren får medvetet utnyttja kunskaper som inhämtats tidigare i appen: Finns det en progression i appen som kräver att användaren reflekterar över vad hen har lärt sig under tidigare banor/nivåer/övningar, och utnyttjar dessa kunskaper för att ta sig vidare? Exempel: under en övning lär sig användaren med hjälp av instruktioner hur man programmerar en slinga. I nästa övning finns bara ett begränsat utrymme att skriva sin programmering på, vilket kräver att användaren gör en slinga för att programmeringen ska få plats. Användaren behöver då först lista ut att en slinga behövs, sedan var slingan ska placeras i programmeringen och slutligen tänka tillbaka hur man programmerar en slinga.

Stegvisa instruktioner

Användaren får följa stegvisa instruktioner: Får användaren själv följa en stegvis instruktion? Exempel: användarens robot står på start. Användaren ska sedan följa en instruktion steg för steg för att lista ut var målet är. Eller, användaren får reda på vart målet finns och ska med hjälp av en stegvis instruktion lista ut var roboten ska starta.

Användaren får skapa egna stegvisa instruktioner: Får användaren skapa stegvisa instruktioner? Exempel: användarens robot står på start och ska ta sig till mål. Användaren får då skapa en programmering i form av en stegvis instruktion för att roboten ska gå till målet.

Pedagogisk utformning

Denna del av analysmodellen handlar om hur apparna i sin helhet lämpar sig för att användas i undervisning på lågstadiet. Analysområdena och kriterierna har valts utifrån vad lärare anser vara viktigt vid val av app till undervisning, baserat på Walkers (2011) analysverktyg för appar.

Användarvänlighet

Lättförstådda instruktioner: Är det enkelt att förstå vad man som användare förväntas göra i appen? Exempel: instruktionerna skrivs med enkel text, instruktionerna läses upp och då i ett lagom tempo, instruktionerna ges visuellt genom att appen visar hur man genomför nya moment, nya begrepp beskrivs.
Exempel på ej uppfyllt kriterium: instruktionerna är endast på engelska, har ett för högt tempo, består av mycket text, saknar förklaringar av nya begrepp, eller instruktioner saknas helt.

Möjligt att välja språk: Är det möjligt att ändra språk i appen? Exempel: ändra talat språk, ändra skrivet språk.

Möjligt att pröva sig fram: Är det möjligt att pröva sig fram till rätt lösning? Exempel: instruktionerna är på ett språk man som användare inte förstår, det är då möjligt att pröva sig fram till förståelse.

Differentiering

Möjligt att nivåanpassa: Är det möjligt att nivåanpassa appen åt eleverna? Exempel: att programmera slingor är för svårt, eleven kan då gå tillbaka och träna grunderna. Att programmera slingor är för enkelt, eleven kan då gå vidare och träna villkorssatser. Att programmera med blockbaserad programmering är för enkelt, eleven kan då gå vidare och

programmera med textbaserad programmering. Läraren kan via inloggning på internet tilldela eleverna uppgifter.

Möjligt att träna en viss färdighet: Är det möjligt för användaren att välja en viss färdighet att öva på? Exempel: appen innehåller banor, nivåer eller övningar där man specifikt tränar grunderna (sekvensering), slingor, villkorssatser, osv.

Feedback

Eleven får feedback om sitt eget resultat: Får användaren reda på hur hen har lyckats med uppgifterna i appen? Exempel: användaren får belöningar i form av poäng, pengar, stjärnor, troféer, m.m.

Läraren får feedback om elevens resultat: Kan läraren får reda på elevernas resultat utan att behöva kolla varje elevs surfplatta, inloggning eller dylikt? Exempel: läraren kan via inloggning på internet skapa en sida för sin klass där hen kan se elevernas resultat.

Bilaga 2

Analys av appar

I följande bilaga finns analyserna av samtliga 14 appar i alfabetisk ordning.

A.L.E.X. (version 1.64) Roboten A.L.E.X. ska i olika banor programmeras så den steg för steg tar sig från start till mål. Användaren kan också skapa egna banor åt A.L.E.X. Användaren får möta sekvensering.	
Matematiskt innehåll	**Kriterier**
Mönster	Användaren får skapa egna mönster
	Användaren får resonera kring mönsters uppbyggnad
Symboler	Användaren får möta olika typer av symboler
	Användaren får förståelse för olika symbolers betydelse
	Användaren får skapa med symboler
Generaliseringar	Användaren måste medvetet utnyttja kunskaper som inhämtats tidigare i appen
Stegvisa instruktioner	
	Användaren får skapa egna stegvisa instruktioner
Pedagogisk utformning	**Kriterier**
Användarvänlighet	Lättförstådda instruktioner
	Möjligt att pröva sig fram
Differentiering	
Feedback	Eleven får feedback om sitt eget resultat

Bee-Bot (version 1.28) Ett bi programmeras att steg för steg ta sig från start till mål. Användaren samlar poäng, och det går på tid. Ju snabbare användaren anger rätt programmering, desto högre poäng. Användaren får möta sekvensering.	
Matematiskt innehåll	**Kriterier**
Mönster	
Symboler	Användaren får möta olika typer av symboler
	Användaren får förståelse för olika symbolers betydelse
	Användaren får skapa med symboler
Generaliseringar	Användaren måste medvetet utnyttja kunskaper som inhämtats tidigare i appen
Stegvisa instruktioner	
	Användaren får skapa egna stegvisa instruktioner
Pedagogisk utformning	**Kriterier**
Användarvänlighet	Lättförstådda instruktioner
	Möjligt att pröva sig fram
Differentiering	
Feedback	Eleven får feedback om sitt eget resultat

BlueBot (version 1.15) En robot ska programmeras att ta sig steg för steg från start till mål. Användaren kan välja mellan att följa en färdig programmering eller skapa en egen. Användaren får möta sekvensering.	
Matematiskt innehåll	**Kriterier**
Mönster	Användaren får skapa egna mönster
	Användaren får resonera kring mönsters uppbyggnad
Symboler	Användaren får möta olika typer av symboler
	Användaren får förståelse för olika symbolers betydelse
	Användaren får skapa med symboler
Generaliseringar	Användaren måste medvetet utnyttja kunskaper som inhämtats tidigare i appen
Stegvisa instruktioner	Användaren får följa stegvisa instruktioner
	Användaren får skapa egna stegvisa instruktioner
Pedagogisk utformning	**Kriterier**
Användarvänlighet	Lättförstådda instruktioner
	Möjligt att pröva sig fram
Differentiering	Möjligt att nivåanpassa
	Möjligt att träna en viss färdighet
Feedback	Eleven får feedback om sitt eget resultat

Box Island (version 2.2.0) Användaren ska programmera en figur att steg för steg gå en bana och samla stjärnor. Svårighetsgrad ställs in efter ålder. Användare 6-11 år möter sekvensering, slingor och villkor.	
Matematiskt innehåll	**Kriterier**
Mönster	Användaren får skapa egna mönster
	Användaren får resonera kring mönsters uppbyggnad
Symboler	Användaren får möta olika typer av symboler
	Användaren får förståelse för olika symbolers betydelse
	Användaren får skapa med symboler
Generaliseringar	Användaren måste medvetet utnyttja kunskaper som inhämtats tidigare i appen
Stegvisa instruktioner	
	Användaren får skapa egna stegvisa instruktioner
Pedagogisk utformning	**Kriterier**
Användarvänlighet	Lättförstådda instruktioner
	Möjligt att välja språk
	Möjligt att pröva sig fram
Differentiering	Möjligt att nivåanpassa
Feedback	Eleven får feedback om sitt eget resultat

Coda Game (version 1.4) Användaren får lära sig att programmera egna spel. Spelen kan sedan delas med andra, och man kan spela spel som gjorts av andra användare.	
Matematiskt innehåll	**Kriterier**
Mönster	
Symboler	Användaren får möta olika typer av symboler
	Användaren får förståelse för olika symbolers betydelse
	Användaren får skapa med symboler
Generaliseringar	
Stegvisa instruktioner	
Pedagogisk utformning	**Kriterier**
Användarvänlighet	Lättförstådda instruktioner
	Möjligt att välja språk
	Möjligt att pröva sig fram
Differentiering	
Feedback	

CodeSpark Academy: The Foos (version 2.16.01) Användaren ska programmera figurer steg för steg att lösa problem, och får då öva sekvensering, slingor, händelser och villkorssatser. I spelskaparläge får användaren skapa egna spel, och spela spel gjorda av andra användare. Läraren har möjlighet att ge eleverna individuella inloggningar.	
Matematiskt innehåll	**Kriterier**
Mönster	Användaren får skapa egna mönster
	Användaren får resonera kring mönsters uppbyggnad
Symboler	Användaren får möta olika typer av symboler
	Användaren får förståelse för olika symbolers betydelse
	Användaren får skapa med symboler
Generaliseringar	Användaren måste medvetet utnyttja kunskaper som inhämtats tidigare i appen
Stegvisa instruktioner	
	Användaren får skapa egna stegvisa instruktioner
Pedagogisk utformning	**Kriterier**
Användarvänlighet	Lättförstådda instruktioner
	Möjligt att välja språk
	Möjligt att pröva sig fram
Differentiering	Möjligt att nivåanpassa
	Möjligt att träna en viss färdighet
Feedback	Eleven får feedback om sitt eget resultat
	Läraren får feedback om elevens resultat

Hopscotch (version 3.26.4) Användaren får med hjälp av blockbaserad programmering skapa egna spel.	
Matematiskt innehåll	**Kriterier**
Mönster	
Symboler	
Generaliseringar	Användaren måste medvetet utnyttja kunskaper som inhämtats tidigare i appen
Stegvisa instruktioner	
	Användaren får skapa egna stegvisa instruktioner
Pedagogisk utformning	**Kriterier**
Användarvänlighet	
	Möjligt att pröva sig fram
Differentiering	
Feedback	

Kodable (version 7.5.0) Användaren ska programmera en prick att rulla från start till mål genom att ge instruktioner om vilket håll pricken ska rulla. Användaren får möta sekvensering och slingor. I skolläge har alla elever individuella inloggningar.	
Matematiskt innehåll	**Kriterier**
Mönster	Användaren får skapa egna mönster
	Användaren får resonera kring mönsters uppbyggnad
Symboler	Användaren får möta olika typer av symboler
	Användaren får förståelse för olika symbolers betydelse
	Användaren får skapa med symboler
Generaliseringar	Användaren måste medvetet utnyttja kunskaper som inhämtats tidigare i appen
Stegvisa instruktioner	
	Användaren får skapa egna stegvisa instruktioner
Pedagogisk utformning	**Kriterier**
Användarvänlighet	Lättförstådda instruktioner
	Möjligt att pröva sig fram
Differentiering	Möjligt att nivåanpassa
	Möjligt att träna en viss färdighet
Feedback	Eleven får feedback om sitt eget resultat
	Läraren får feedback om elevens resultat

LEGO: Fix the Factory (version 1.4.2) En robot ska programmeras att steg för steg ta sig igenom olika banor i en fabrik. Användaren får möta sekvensering.	
Matematiskt innehåll	**Kriterier**
Mönster	Användaren får skapa egna mönster
	Användaren får resonera kring mönsters uppbyggnad
Symboler	Användaren får möta olika typer av symboler
	Användaren får förståelse för olika symbolers betydelse
	Användaren får skapa med symboler
Generaliseringar	Användaren måste medvetet utnyttja kunskaper som inhämtats tidigare i appen
Stegvisa instruktioner	Användaren får skapa egna stegvisa instruktioner
Pedagogisk utformning	**Kriterier**
Användarvänlighet	Lättförstådda instruktioner
	Möjligt att pröva sig fram
Differentiering	
Feedback	Eleven får feedback om sitt eget resultat

Lightbot: Code Hour (version 1.6.8) En robot ska programmeras att steg för steg ta sig igenom olika banor och tända lampor på marken. Användaren får möta sekvensering och slingor.	
Matematiskt innehåll	**Kriterier**
Mönster	Användaren får skapa egna mönster
	Användaren får resonera kring mönsters uppbyggnad
Symboler	Användaren får möta olika typer av symboler
	Användaren får förståelse för olika symbolers betydelse
	Användaren får skapa med symboler
Generaliseringar	Användaren måste medvetet utnyttja kunskaper som inhämtats tidigare i appen
Stegvisa instruktioner	
	Användaren får skapa egna stegvisa instruktioner
Pedagogisk utformning	**Kriterier**
Användarvänlighet	
	Möjligt att välja språk
	Möjligt att pröva sig fram
Differentiering	
	Möjligt att träna en viss färdighet
Feedback	Eleven får feedback om sitt eget resultat

Pyonkee (version 2.6) Användaren programmerar korta filmer och berättelser med det blockbaserade programmerings-språket Scratch.	
Matematiskt innehåll	**Kriterier**
Mönster	
Symboler	
Generaliseringar	Användaren måste medvetet utnyttja kunskaper som inhämtats tidigare i appen
Stegvisa instruktioner	
	Användaren får skapa egna stegvisa instruktioner
Pedagogisk utformning	**Kriterier**
Användarvänlighet	
	Möjligt att välja språk
	Möjligt att pröva sig fram
Differentiering	
Feedback	

Scratch Jr (version 1.2.5) Användaren får skapa korta filmer och berättelser med det blockbaserade programmeringsspråket Scratch.	
Matematiskt innehåll	**Kriterier**
Mönster	
Symboler	Användaren får möta olika typer av symboler
	Användaren får förståelse för olika symbolers betydelse
	Användaren får skapa med symboler
Generaliseringar	Användaren måste medvetet utnyttja kunskaper som inhämtats tidigare i appen
Stegvisa instruktioner	
	Användaren får skapa egna stegvisa instruktioner
Pedagogisk utformning	**Kriterier**
Användarvänlighet	
	Möjligt att välja språk
	Möjligt att pröva sig fram
Differentiering	
	Möjligt att träna en viss färdighet
Feedback	

Swift Playgrounds (version 2.0) Användaren lär sig att programmera med det textbaserade programmeringsspråket Swift. Användaren får först lära sig grunderna, och kan sedan programmera egna spel i appen. Användaren sekvensering, slingor, villkor, buggar och funktioner.	
Matematiskt innehåll	**Kriterier**
Mönster	Användaren får skapa egna mönster
	Användaren får jämföra mönster
	Användaren får resonera kring mönsters uppbyggnad
Symboler	
Generaliseringar	Användaren måste medvetet utnyttja kunskaper som inhämtats tidigare i appen
Stegvisa instruktioner	
	Användaren får skapa egna stegvisa instruktioner
Pedagogisk utformning	**Kriterier**
Användarvänlighet	
	Möjligt att pröva sig fram
Differentiering	Möjligt att nivåanpassa
	Möjligt att träna en viss färdighet
Feedback	Eleven får feedback om sitt eget resultat

Tynker (version 4.4.50) Appen innehåller två spel: det ena handlar om en astronaut och det andra om en drake. I båda spelen ska användaren programmera figurerna att steg för steg ta sig igenom olika banor. Användaren får välja mellan blockbaserad eller textbaserad programmering. Användaren möter sekvensering, slingor, villkor, buggar och funktioner.	
Matematiskt innehåll	**Kriterier**
Mönster	Användaren får skapa egna mönster
	Användaren får resonera kring mönsters uppbyggnad
Symboler	
Generaliseringar	Användaren måste medvetet utnyttja kunskaper som inhämtats tidigare i appen
Stegvisa instruktioner	
	Användaren får skapa egna stegvisa instruktioner
Pedagogisk utformning	**Kriterier**
Användarvänlighet	Lättförstådda instruktioner
	Möjligt att välja språk
	Möjligt att pröva sig fram
Differentiering	Möjligt att nivåanpassa
	Möjligt att träna en viss färdighet
Feedback	Eleven får feedback om sitt eget resultat