PROGRAMMERA MERA – TECKENSPRÅK

LÄRARHANDLEDNING

FÖRFATTARE
KARIN NYGÅRDS

PROGRAMMERA MERA

Vi lever i en digitaliserad värld och för barn idag är den digitala världen självklar. För att förstå att det är en värld skapad av oss människor, och för att få barn att vara kritiska producenter och medvetna konsumenter, krävs kunskap om programmering.

ETT LUSTFYLLT PROGRAM OM PROGRAMMERING

Serien *Programmera mera* vill på ett lustfyllt sätt berätta och visa hur programmering fungerar. Den nioåriga tittaren som är nybörjare inom programmering får hänga med genom tio fartfyllda avsnitt där tävlande barn, under ledning av programledaren Karin Nygårds, möter utmaningar kopplade till programmering. Det handlar om logiskt tänkande, sortering och om att se mönster och samband. I det animerade historiska inslaget får vi veta mer om allt från vem som uppfann den första datorn till hur Internet startade.

PROGRAMMERING – ETT SÄTT ATT TÄNKA

Fokus i serien är tankesättet kring programmering snarare än att faktiskt lära sig skriva kod eller kunna programmeringsspråk. Det är inspiration till elever och pedagoger som vill lära sig grunderna i programmering utan att behöva förkunskaper eller teknisk utrustning. Svårigheten i utmaningarna ökar efterhand vilket bidrar till progression i serien.

TV-PROGRAM, LEKTIONSTIPS OCH HANDLEDNING

Till serien hör även tio filmer på cirka tre minuter vardera, *Programmera mera – Lektionstips*, som beskriver begrepp och visar praktiskt hur en del av övningarna som finns beskrivna i handledningen kan göras i klassrummet. Om man är helt obekant med programmering, kan det vara en bra start att börja med att titta på webbfilmerna som du hittar på urskola.se.
Programmering är inte bara kopplat till matematik och teknik utan i lika hög grad till ämnen som idrott och hälsa, svenska och samhällsorientering. I den här handledningen får du förslag som kan anpassas till olika ämnen.

TACK TILL

Utmaningarna i programmet och övningarna i handledningen är framtagna i samråd med Design Technologist Ellen Sundh, professor Peter Parnes och docent Fredrik Heintz. Stort tack även till Terese Raymond som suttit med i referensgrupp samt skrivit de historiska bakgrundstexterna i handledningen. Programmeringstävlingen bebras.se. har fungerat som inspiration och för den pedagogiska expertisen står programledaren Karin Nygårds som har utformat övningarna utifrån sin egen fleråriga undervisning inom programmering.

VAD SÄGER STYRDOKUMENTEN?

Tankesättet inom programmering är en tillgång i många av skolans ämnen. Men än viktigare är att kunskap om programmering och tillgång till de förmågor som behövs för att programmera är en förutsättning för individens möjlighet att vara en delaktig i det samhälle skolan har ansvar att förbereda elever för.

”Skolan ska ansvara för att varje elev efter genomgången grundskola
- kan använda sig av matematiskt tänkande för vidare studier och i vardagslivet,
- kan lösa problem och omsätta idéer i handling på ett kreativt sätt,
- kan lära, utforska och arbeta både självständigt och tillsammans med andra och känna tillit till sin egen förmåga,
- kan göra väl underbyggda val av fortsatt utbildning och yrkesinriktning.”

(ur Lgr 11, 2. Övergripande mål och riktlinjer, 2.2 Kunskaper)

I både syftesbeskrivningar och det centrala innehållet till flera ämnen finns formuleringar där kopplingen till kunskap om, och färdighet i, programmering syns självklar.

MATEMATIK

”Genom undervisningen ska eleverna ges förutsättningar att utveckla förtrogenhet med grundläggande matematiska begrepp och metoder och deras användbarhet. Vidare ska eleverna genom undervisningen ges möjligheter att utveckla kunskaper i att använda digital teknik för att kunna undersöka problemställningar, göra beräkningar och för att presentera och tolka data.”

Centralt innehåll
I årskurs1-3
• Hur enkla mönster i talföljder och enkla geometriska mönster kan konstrueras, beskrivas och uttryckas.

TEKNIK

Syfte
”Undervisningen ska bidra till att eleverna utvecklar kunskaper om teknikens historiska utveckling för att de på så sätt bättre ska förstå dagens komplicerade tekniska företeelser och sammanhang och hur tekniken påverkat och påverkar samhällsutvecklingen.”

Centralt innehåll
I årskurs 1-3
- Några föremål i elevens vardag och hur de är anpassade efter människans behov.
- Hur föremålen i elevens vardag har förändrats över tid.

MUSIK

Syfte
”Undervisningen ska ge eleverna förutsättningar att tillägna sig musik som uttrycksform och kommunikationsmedel. Genom undervisningen ska eleverna ges möjlighet att utveckla kunskap att använda röst, musikinstrument, digitala verktyg samt musikaliska begrepp och symboler i olika musikaliska former och sammanhang.”

Centralt innehåll
I årskurs 1-3
- Enkla former av musikskapande, till exempel med utgångspunkt i text eller bild.

Också den undervisning som fritidshemmet ansvarar för har tydlig koppling till programmering.

FRITIDSHEMMET

Syfte
Genom undervisningen i fritidshemmet ska eleverna sammanfattningsvis ges förutsättningar att utveckla sin förmåga att
- pröva och utveckla idéer, lösa problem och omsätta idéerna i handling,
- utforska och beskriva företeelser och samband i natur, teknik och samhälle,

Centralt innehåll
Undervisningen ska behandla följande centrala innehåll

SPRÅK OCH KOMMUNIKATION
- Digitala verktyg och medier för kommunikation
- Säker och ansvarsfull kommunikation, även i digitala sammanhang

SKAPANDE OCH ESTETISKA UTTRYCKSFORMER
- Digitala verktyg för framställning av olika estetiska uttryck.

NATUR OCH SAMHÄLLE
- Olika sätt att utforska företeelser och samband i natur, teknik och samhälle, till exempel genom samtal, studiebesök och digitala medier.
- Hur företeelser och samband kan beskrivas, till exempel med ord och bilder.
- Matematik som redskap för att beskriva vardagliga företeelser och för att lösa vardagliga problem.

KORT OM SERIEN - INSLAG I VARJE AVSNITT

DELTAGARNA PRESENTERAR SIG

Varje avsnitt inleds med att deltagarna presenterar sig. I studion kopplas några av deras intressen ihop med programmering. Se detta som tips på en introduktion till vad programmering är och använd liknande kopplingar för att aktivera elevernas funderingar och nyfikenhet.

UPPGIFTER

Deltagarna får lösa tre uppgifter på egen hand, utan programledarens hjälp. Tanken är att de ska samarbeta och våga prova. Tävlingsformen i programmen är vald för att öka spänningen för tittaren, men är inget som är nödvändigt att använda sig av i klassrummet. Den korta programtiden gör att deltagarna inte hinner försöka flera gånger, vilket annars är en viktig del av programmering. I klassrummet bör eleverna uppmuntras att prova om och om igen, tills de lyckas.

Utmaningarna speglar förmågor och tar upp begrepp som är viktiga utifrån olika aspekter av programmering.

HISTORISKT INSLAG

Varje avsnitt innehåller ett kort animerat inslag, vars syfte är att sätta in programmering och datorer i ett historiskt sammanhang. Programmering har funnits i vårt samhälle under lång tid och det har på många sätt påverkat den värld vi lever i.

AVSLUTANDE FRÅGA

När deltagarna har genomfört sina tre uppgifter kommer de tillbaka till programledaren som väntar med en sista fråga som berör något de haft möjlighet att lära sig under besöket i studion. Här kan du som lärare pausa och låta eleverna i klassen svara på frågan, innan ni får facit av programledaren.

BRA ATT KÄNNA TILL - PROGRAMMERINGSBEGREPP

ALGORITM

I matematikundervisningen möter eleverna ofta algoritmer. Då brukar det handla om uppställningar av tal. Ordet betyder precisa instruktioner för att utföra en uträkning och det är ett centralt begrepp inom programmering. För att få datorn att göra som vi vill, måste vi ge den mycket noggranna instruktioner. En dator kan inte läsa mellan raderna, hoppa mellan instruktioner eller tolka och förstå utifrån sammanhang.
Ett sätt att visa skillnaden mellan en dator och en människas sätt att förstå instruktioner är att ge eleverna en uppmaning som ”slå upp matteboken och fortsätt jobba”. Här förstår eleverna förhoppningsvis vad de ska göra. De ska gå och hämta matteboken, öppna den på den sida där de är och arbeta vidare. De förstår också att de ska ta med sig penna och sudd och annat de behöver för att kunna utföra arbetet. En dator hade fastnat direkt. Om vi låtsas att datorn förstod vårt språk skulle den läsa så här:

```
slå          –okej, jag ska slå någon. Vem ska jag slå?
upp          – jag ska slå uppåt
matteboken   – vad betyder det? Vad ska jag göra? Nu förstår jag inte. SYNTAX ERROR
```

En dator behöver mycket mer precisa och exakta instruktioner än vad vi människor gör . Det kan vara svårt för oss att tänka så som datorer fungerar, för vi tar så mycket för givet. Att allt måste komma i rätt ordning kan vara utmanande när man programmerar.

I Programmera mera presenteras begreppet algoritm när deltagarna ska sätta instruktioner i rätt ordning. Instruktionerna står på remsor som ska fästas på en tavla.

Det är tacksamt att jobba med algoritmer kopplat till hur man skriver instruktioner. Våra dagar är fulla av algoritmer som vi följer, utan att vi ens tänker på det. Klassens schema är ett exempel på en daglig algoritm.

Att kunna ge tydliga instruktioner är något man har nytta av i många situationer och därför behöver träna på. En träning som lätt går att koppla ihop med ämnen som svenska, engelska, främmande språk, idrott och hälsa eller SO.

ATT STYRA MED PROGRAMMERING

I varje avsnitt finns en uppgift som handlar om att styra med hjälp av programmering. Det är ett bra sätt att närma sig vad programmering handlar om. Idag finns det flera olika slags robotar som är rimliga i pris för den som vill köpa in till skolan; men det går faktiskt bra att få en förståelse för hur man styr med programmering genom att styra sin kompis eller lärare också.

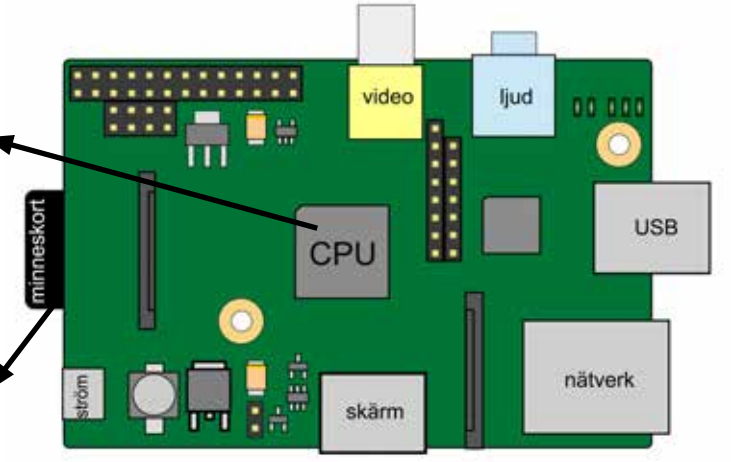
Programmets mänskliga robot styrs med hjälp av ett påhittat kodspråk bestående av verb som *gå, snurra, ta, vänd* och substantiv som *arm, ben, blomma* och så vidare. Alla ord står på färgade plastbrickor som deltagarna måste lägga i rätt ordning. Svårigheten ligger i att tänka efter hur de olika rörelserna ska göras, innan man testar. När man programmerar måste man nästan alltid pröva många gånger innan det blir rätt. Det hade inte deltagarna i programmet möjlighet till, eftersom det hade tagit för lång tid. Men att göra skisser och planer, och att prova sig fram, är nyttiga verktyg för att styra med hjälp av programmering.

VAD ÄR PROGRAMMERAT?

Ett syfte i ämnet teknik är att eleverna utvecklar sitt tekniska kunnande och sin tekniska medvetenhet, så att de kan orientera sig och agera i en teknikintensiv värld. För att kunna göra det kan vi sätta fokus på vad som är programmerat eller inte. Det är inte helt självklart vad som gör ett föremål programmerbart. Vad är ens en dator? En dator kan se ut på många olika sätt. Det finns allt från superdatorer som är flera hundra kvadratmeter stora, till microprocessorer som kan vara så små som några millimeter.

Processorn är den del i datorn som hanterar all data. Den kallas CPU (central processing unit). CPU läser in ettor och nollor som kommer in till datorn, detta kallas indata.

Minnet är den del där program och data sparas. CPU kollar med minnet hur ettorna och nollorna ska hanteras.



När CPU har tolkat indata, skickar den ut utdata, alltså det som efterfrågades. Det kan vara text, bild, musik, film eller något kommando som användaren vill att datorn ska utföra.

I programmet finns hyllor med olika saker, där de medverkande väljer ut tre föremål med programmerad data. Hur ser det ut i ert klassrum? Hur många saker är programmerade där? Exempel på föremål som kan finnas i klassrum: *dator, surfplattor, projektor, miniräknare, klocka* (finns både digitala och analoga, vilket inte eleverna alltid vet skillnaden mellan), *digital termometer, tidtagarur, digital våg, digitalpiano, larm, högtalarsystem, rastklocka*. Hemma har vi antagligen ännu fler programmerade saker, till exempel: *tvättmaskin, diskmaskin, kaffekokare, micro, TV, fjärrkontroll, telefon, laddare, brandvarnare, leksaker, verktyg, tandborste*.

MASKINKOD

Maskinkod är det som själva datorn läser. Den kallas även binärkod, eftersom den bara består av två lägen, ett eller noll. I grunden kan datorer bara förstå ettor och nollor, eller elektriska impulser som är på eller av.

För att datorn ska förstå vad ettorna och nollorna betyder, används olika system för kodning. Genom att tala om för datorn på vilket sätt maskinkoden ska tolkas, kan signalerna bilda text, bild eller film.

I serien används en tabell som heter ASCII, American Standard Code for Information Interchange. I ASCII-tabellen motsvarar åtta ettor och nollor i en bestämd kombination en bestämd bokstav.

Det går inte att skriva ÅÄÖ med ASCII. I dag använder man hellre UNICODE som består av 32 ettor och nollor som kan bilda många fler kombinationer. Med UNICODE kan vi använda ÅÄÖ i webbadresser, därför är den bättre. Men när man ska lära sig något om maskinkod är det jobbigare att använda UNICODE eftersom det är ett mer komplicerat system än ASCII.

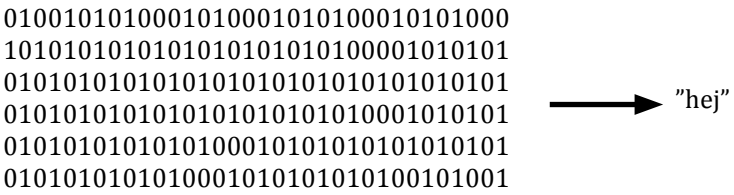
I serien används en specialbyggd tavla som har tre rader med åtta glödlampor i varje rad. Varje rad symboliserar en bokstav. Lamporna tänds eller får vara släckta, beroende på om de ska markera en etta eller en nolla – eller med andra ord på eller av.

0100 0001	A
0100 0010	B
0100 0011	C
0100 0100	D
0100 0101	E
0100 0110	F
0100 0111	G
0100 1000	H
0100 1001	I
0100 1010	J
0100 1011	K
0100 1100	L
0100 1101	M
0100 1110	N
0100 1111	O
0101 0000	P
0101 0001	Q
0101 0010	R
0101 0011	S
0101 0100	T
0101 0101	U
0101 0110	V
0101 0111	W
0101 1000	X
0101 1001	Y
0101 1010	Z

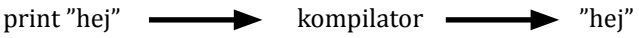
KOMPILATOR

Datorn förstår bara maskinkod som består av ettor och nollor. Om man skulle programmera en dator med hjälp av maskinkod skulle vi behöva skriva flera miljarder ettor och nollor. Det slipper man tack och lov. Grace Hopper (se mer om henne i det historiska inslaget i avsnitt 4) uppfann kompilatorn, som översätter bokstäver till maskinkod. Det gör att man kan använda programmeringsspråk istället för maskinkod när man programmerar.

Utan kompilator:



Med kompilator:



PROGRAMMERINGSSPRÅK

Idag finns fler än tusen olika programmeringsspråk. Anledningen till att det finns så många är att de används till olika saker. Företag kan ta fram egna språk för att programmera en viss slags robot i en fabrik till exempel. Dessutom sker det hela tiden utveckling och avknoppningar av program, som blir till nya språk, till exempel C som har blivit C# (uttalas ”si sharp”) och C++.

I serien visas kod skrivet i JavaScript, C++ och Python. Även programmeringsspråket Scala förekommer, översatt till svenska. Deltagarna får i två avsnitt använda så kallad blockprogrammering, som är speciellt framtaget för att lära barn programmera. Istället för att skriva kod, dras block ihop som pusselbitar och talar om för datorn vad den ska utföra.

MÖNSTER

Förmågan att kunna se och upprepa mönster är bra att träna för att bli duktig på att programmera. När man programmerar är det en fördel om man kan se på vilka sätt kod återkommer, så att man slipper skriva samma sak flera gånger. I början spelar det inte så stor roll, men ju mer komplicerade program man skriver, desto viktigare blir det att koden blir lättläst och inte för lång.

Exempel:
Uppgiften att skriva ett program som skriver ut en kvadrat kan se ut på olika sätt.

```
fram
höger
fram
höger
fram
höger
fram
höger
```

```
upprepa(4) {fram;höger}
```

Det är inte alltid så lätt att hitta mönster för att skriva bra program. Det är något som är nyttigt att träna och att påminna eleverna om när de jobbar med programmering. Till exempel: om eleverna har lärt sig göra en kvadrat som ovan, kan nästa uppgift vara att göra ett trappmönster.

upprepa(4) {fram;höger;fram;vänster}
Det är också lättare att hitta fel, eller buggar som det brukar kallas, i programmeringskod, om man är bra på att se mönster eftersom man då också ser var en avvikelse finns.

LOOP

En loop är något som upprepas flera gånger. I texten här ovan, som handlar om mönster, används en loop för att förkorta koden. I stället för att skriva fram, höger, fram, höger och så vidare, används upprepa(4). Loopar kan skrivas på olika sätt men i många programmeringsspråk används *måsvingar*, { }, även kallade *krullparenteser*, för att tala om vad det är som ska upprepas. upprepa(4){fram,höger}

I serien använder deltagarna en loop när roboten ska göra en tårta. Roboten ska ta en tårtbotten, lägga på sylt, ta en tårtbotten till och lägga på sylt igen. Ett annat tacksamt sätt att presentera begreppet loop är att använda musik, där det ofta förekommer upprepningar i till exempel refrängen.

SORTERING

Tänk på hur många olika sätt vi kan sortera elever i en klass på: *alfabetisk ordning, längd, vikt, kön, födelsedatum, hårfärg, husdjur, skostorlek*. Om vi ska skriva instruktioner för att sortera eleverna efter dessa olika sätt, behövs olika slags instruktioner. Att göra en klasslista efter elevernas vikt vore otroligt bökigt och inte särskilt beständigt. På samma sätt måste programmerare tänka ut hur data ska sorteras på bästa sätt i olika situationer. Datorer har alltid använts till sortering. De är oslagbart snabba på att behandla stora mängder data och hitta struktur och ordning. Men för att kunna sortera effektivt måste datorn programmeras med rätt instruktioner, eller algoritmer som det kallas.

I Programmera mera finns ett skåp med glasögon, klossar och varor som ska sorteras efter instruktioner. Det går så klart att göra på liknande sätt med föremål i klassrummet, eller varför inte ute i naturen?

VILLKOR

Mycket av det som händer när vi använder datorer, styrs av villkor. Det betyder att programmet i datorn ska agera olika, beroende på vilka villkor som uppfylls. Ett elevnära exempel är datorspel. De är fyllda av villkor. Om en figur trillar i vattnet, kommer något att hända. Om en figur springer på en skatt, händer något annat. Om poängen blir högre än 100, har spelaren vunnit. Om poängen blir under 0, har spelaren förlorat.

Villkor finns det också gott om i tjänster vi använder på nätet. En sådan diskussion som eleverna kanske känner till är googles algoritmer för att visa våra sökresultat. De är beroende av villkor som programmerare har skrivit. Om en användare tidigare har sökt på katter, kommer sökresultat för ordet ”husdjur” främst visa katter. Googles algoritmer är förstås mer avancerade än så, och dessutom hemliga, men det är ett exempel som de flesta kan relatera till. Många elever är också bekanta med de förslag de får på olika filmtjänster som exempelvis youtube. Om de tidigare har sett en viss typ av filmer, får de nästa gång tips om liknande filmer.

I programmet möter deltagarna villkor när de sorterar efter instruktioner. Bland annat ska de sortera föremål efter vad användare på nätet redan har handlat.

LOGISKT TÄNKANDE

Logiskt tänkande tränas ständigt vid programmering. Ofta påverkar en ändring på ett ställe, hela programmet och ibland blir allt fel på grund av en liten justering. Tänk bara på alla app-uppdateringar till exempel. Så fort en bugg har fixats på ett ställe, verkar det som om nya uppstår och snart är det dags för uppdatering igen. Digitala system är komplexa och det är inte alltid så lätt att hålla reda på hur en ändring påverkar andra delar.

Att tänka i flera led och försöka komma fram till hur ett problem ska lösas bit för bit, är inte bara användbart i många skolämnen och vardagssituationer, det är också en central del i programmering.

I programmet finns en vägg med vattenslangar som öppnas med kranar. Vissa slangar leder ner i en tom burk, medan andra leder till burkar med en mobiltelefon i. Det gäller att öppna rätt kranar, så att det inte rinner ner vätska på mobilerna.

HITTA BUGGEN

Ibland sägs det att det är lika viktigt att kunna läsa kod, som att kunna skriva den. En anledning till att det är bra att kunna läsa kod och förstå vad den ska göra, är för att det blir lättare att hitta fel, när programmet inte fungerar som det ska. Det blir nämligen ofta fel när man programmerar. Antingen tänker man fel, eller så smyger sig felen in när man skriver själva koden. Det är lätt hänt. Fast det är inte alltid lika lätt att hitta vad som är fel! Därför är det bra att träna på att leta fel, eller buggar som det kallas.

En bugg är helt enkelt ett fel i ett program. Namnet bugg kommer från engelskans bug som betyder insekt. Enligt legenden var det en insekt som flög in i en av de tidiga datorerna och orsakade datorhaveri, men troligen är uttrycket så gammalt som 1870-tal, då Edison hade problem med vad han kallade för ”en bugg” i sin fonograf, som han försökte få att fungera.

I programmet fick deltagarna leta buggar i riktiga programmeringsspråk. Det låter svårt, men det fungerar som en klassisk finn fem fel-uppgift. Korrekt kod jämförs med kod där några fel har smugit sig in. Deltagarna hittade felen genom att noggrant jämföra de två versionerna. Man behöver alltså inte kunna ett dugg om programmering, men det är kul att få se hur ”riktig” programmeringskod ser ut.

AVSNITT I: VAD ÄR MASKINKOD?

INNAN PROGRAMMET

Berätta att *datorer* är maskiner som kan samla in, bearbeta och lagra *data*. Data är ett annat ord för information. Datorn styrs av elektriska impulser som brukar skrivas med ettor och nollor och kallas för maskinkod.

UNDER PROGRAMMET

Karin frågar deltagarna om deras intressen och om det går att koppla till programmering på något sätt. Elliot jämför programmering med parkour, där han planerar hur han ska sätta händer och fötter. Det låter kanske långsökt, men planeringen man gör när man ska ta sig från en plats till en annan i parkour, liknar planeringen av hur man ska lösa ett programmeringsproblem och är faktiskt en rätt bra jämförelse.
Deltagarna får varsitt armband som de ska hitta mönstret på. Mönstren på armbanden består av åtta pärlor i två färger. Halsbandet ska ha 40 pärlor, det vill säga fem gånger av samma mönster som på armbandet.
PAUSA! Som slutfråga vill Karin veta vad en ASCII-tabell är för något. Fråga klassen om någon minns vad det är.

EFTER PROGRAMMET

1. Hemligt meddelande (Tips! Titta på Programmera mera – Lektionstips)
Skriv bokstäver och hemliga meddelanden med hjälp av ASCII-tabellen. Du hittar tabellen och övningsblad som kopieringsunderlag 1 och 2. Arbeta enskilt eller i grupp.

- Dela ut kopieringsunderlag 1 och 2.
- Visa hur man fyller i rutorna för att markera (1) på och (0) av.
- Låt alla skriva något med hjälp av ASCII-tabellen och sedan byta med varandra och läsa det hemliga meddelandet.

2. Lysande meddelande (Tips! Titta på Programmera mera – Lektionstips)
Skicka meddelanden med hjälp av ficklampor. Ni behöver:
8 lampor. Helst ficklampor, men andra små lampor går också bra.
Dela ut en ASCII-tabell till varje grupp.

- Dela in er i grupper om åtta. Det går bra med fyra i varje grupp också, då får var och en hålla i två ficklampor var.
- Be grupperna komma på varsitt kort ord. Tre bokstäver kan vara lagom (ha gärna några ord på lager så att inte alla grupper väljer samma ord).
- En grupp i taget går fram, ställer sig på rad och tänder sina ficklampor enligt ASCII-tabellens alfabet.
- De andra grupperna försöker tyda lamporna och leta efter rätt bokstäver i tabellen.
- Upprepa några omgångar så att så många bokstäver som möjligt kommer med. När alla bokstäver är visade, får grupperna redovisa vad de har kommit fram till.

3. För er som vill göra mer
Om eleverna redan är bekanta med ASCII, kan man gå vidare och titta på hur det modernare UNICODE ser ut. Varför kan man skriva fler tecken med UNICODE än med ASCII? (svar: Med 32 ettor och nollor finns många fler alternativ än med bara 8). Det kan vara kul att se att det finns UNICODE-tabeller för nästan alla alfabet, till och med runor! På unicode-table.com kan man hitta alla tecken som finns.

4. Robotar och mönster
Läromedel i matematik brukar ha bra exempel på hur man kan jobba med mönster. Vill man göra kopplingen till programmering tydligare kan man använda begrepp som algoritm och loop för att förtydliga hur det hänger ihop.

5. Konstruera mönster
Låt alla rita eller skriva mönster efter egen fantasi. När de har upprepat sitt mönster en gång byter de med en kompis som ska rita eller skriva vidare enligt samma mönster. Övningen går att utveckla genom att till exempel sammanfoga två elevers mönster och på så vis få en ny mönsterbild att följa.

AVSNITT 2: VAD ÄR PROGRAMMERAT?

INNAN PROGRAMMET

Prata om hur man vet att något är programmerat. Inventera klassrummet. Vilka saker är programmerade? (*dator, surfplattor, projektor, miniräknare, klocka, digital termometer, tidtagarur, digital våg, digitalpiano, larm, högtalarsystem, rastklocka*)

UNDER PROGRAMMET

I avsnittet presenteras en sorteringsövning. Sortering är något som datorer är bra på, men vi måste programmera dem rätt så att sorteringen blir korrekt. Om du till exempel söker något på nätet, får du svaren sorterade i en viss ordning. Programmerare får ofta fundera på hur saker ska sorteras på bästa sätt, därför är det bra att träna på att sortera saker på olika sätt.

Första utmaningen handlar om sortering efter olika villkor. Deltagarna ska sortera glasögon efter instruktioner. PAUSA och titta på villkoren. Kunde villkoren varit några andra?
Den andra utmaningen går ut på att välja ut föremål som är programmerade från en hylla full med olika saker.
PAUSA! Titta på hyllan tillsammans. Vilka saker kan eleverna se som är programmerade?

PAUSA! Som slutfråga vill Karin veta vad sortering har med programmering att göra. Minns ni?

EFTER PROGRAMMET

1. Hur är det sorterat?

Välj ut föremål, till exempel små plastdjur, olika slags pennor, föremål med olika geometriska former. Kanske finns det bra material som tillhör matematiken? Det viktiga är att det finns flera parametrar att sortera efter: färg, form, funktion eller alfabetisk ordning.

- Sortera föremålen på ett sätt. Låt eleverna gissa hur du har tänkt.
- Gör en ny sortering och låt eleverna gissa.
- Fråga om någon kommer på ytterligare något sätt att sortera på.
- Låt eleverna arbeta i mindre grupper på samma sätt med andra föremål.

2. Pilprogrammering

I programmet får deltagarna programmera en bilrobot att ta sig runt i en stad. Roboten programmeras med hjälp av pilar. Det är ett enkelt språk att börja lära sig programmera med. Varje pil framåt, motsvarar ett steg fram. En pil åt höger eller vänster motsvarar en vridning. För att roboten ska ta sig åt höger eller vänster krävs både en svängpil och en pil framåt. Det kan ställa till det ibland.

- Skriv ut Kopieringsunderlag 3. Låt eleverna klippa ut pilarna som de ska använda för att lägga ut koden.
- Använd till exempel knappar eller gem för att symbolisera ”robot” och markera start och stopp på banan.
- Eleverna kan också rita ut en egen värld på pappret om det finns tid till det.
- Eleverna jobbar två och två och lägger ut kod och testar om de tänkt rätt.

3. Förbättra och förändra

Beskriv och berätta om ett programmerat föremål. Hur skulle du vilja förbättra det föremålet? Eller välj föremål som inte är programmerade och kom på hur de skulle kunna programmeras. Rita och skriv för att beskriva.

4. Sorteringsalgoritm (Tips! Titta på Programmera mera – Lektionstips)
Använd Kopieringsunderlag 4.
Datorer programmeras med olika slags algoritmer för att sortera data. Det här är ett exempel på en sorteringsalgoritm. Om den illustreras på golvet kan den användas för att sortera eleverna efter olika parametrar. Det kan vara tacksamt att börja med alfabetisk ordning.

- Lägg ut A4-papper efter mönstret i kopieringsunderlaget. Det går att anpassa och göra större eller mindre om det behövs.
- Sex elever placerar sig i varsin startruta.
- Alla tar ett steg fram så att det står två elever i varje ruta på rad 1.
- Eleverna jämför sina namn. Vem kommer först i alfabetet?
- Den som kommer före i alfabetet går åt höger på bilden, alltså sitt vänster.
- Den som kommer efter i alfabetet, går åt vänster på bilden, alltså sitt höger.
- Nytt par möts och proceduren upprepas.
- Fortsätt tills alla sex eleverna står på varsin målruta.

5. Samla och sortera
Samla föremål med programmerad data; antingen i klassrummet eller ge eleverna i läxa att hitta programmerade föremål. Gör en sammanställning av sakerna. Arbeta gärna med sortering här också – hushållsprodukter i en kolumn, utomhusprylar i en annan och så vidare. Hur är det sorterat?
Vidareutveckla genom att låta eleverna själva välja föremål att sortera.

6. Gör en utmaning
Ordna en egen utmaning i klassrummet, liknande den i programmet, där eleverna ska hitta vad som är programmerat.

AVSNITT 3: ROBOTAR OCH LOOPAR

INNAN PROGRAMMET

Introducera begreppet loop. Loop betyder att något görs om och om igen. Vad har eleverna för loopar i sitt liv? (tandborstning, äta, gå till skolan, läsa ett kapitel i en bok, skärmtid och så vidare).

Prata om vad en robot är. Det som skiljer en robot från en dator är att den även gör något fysiskt. Robotar kan flytta saker, klippa gräs, tvätta kläder, skruva i skruvar och annat tungt och repetitivt arbete. I fabriker finns det ofta fullt med robotar som gör olika saker. Schablonbilden av en robot är den som ser ut som en människa i plåt, men oftast ser de ut som maskiner. Fråga eleverna om de har några robotar hemma. De flesta har åtminstone en tvättmaskin eller en diskmaskin. Vi tänker inte på det som robotar längre, men de är programmerade till att utföra ett fysiskt arbete åt oss.

UNDER PROGRAMMET

PAUSA! I sista uppgiften ska deltagarna i avsnittet programmera roboten så att den kan göra en tårta. Deltagarna måste använda en loop, annars får koden inte plats. Har eleverna någon idé om vad som ska vara med i loopen?

PAUSA! Som slutfråga undrar programledaren varför programmerare använder loopar.

EFTER PROGRAMMET

1. Olika sätt att skriva loopar

Använd Kopieringsunderlag 7 för att visa olika sätt att skriva loopar. Där finns exempel på loopar så som de skrivs i programmeringsmiljöerna code.org, Scratch och Kojo, som alla är anpassade för barn som ska lära sig programmering.

2. Dansprogrammering

Använd Kopieringsunderlag 8 med rörelsekort eller hitta på egna symboler för olika rörelser.

- Börja med att introducera rörelserna.
- Sätt ihop rörelserna till en dans genom att sätta symbolerna i en lodrät rad.
- Dansa igenom rörelserna tillsammans. Varje rörelse upprepas 3 gånger.
- Introducera loop-tecknet och välj en rörelse som ska upprepas.
- Introducera måsvingarna och välj flera rörelser som ska upprepas.

3. Programmera varandra i klassrummet.

Det här är en bra övning för att få syn på grundproblemen när man tar sig an att styra med programmering. Det blir mest effektivt om eleverna själva får upptäcka när koden inte fungerar som de har tänkt sig. Vanliga problem som brukar uppstå är att koden är för otydlig, hoppar över viktiga förklaringar eller att mått som ”ett steg” inte är definierat. Det ger anledning att prata om vikten av att vara precis och exakt och om hur praktiskt det är med det matematiska språket som vi har gemensamt.

- Gruppera eleverna på lämpligt sätt, förslagsvis två och två.
- Eleverna ska skriva kod till varandra för att ta sig från en plats i klassrummet till en annan.
- Det kan bli lite stökigt när alla går runt och räknar steg och testar, men det gäller att ha is i magen, för det brukar vara väldigt roligt. Lämplig tid för att låta eleverna jobba med kodskapandet är ca 10 minuter.
- Prova kod. Eleverna eller grupperna byter kod med varandra och försöker lösa uppgiften.
- Sammanfatta hur det gick för de olika grupperna. Vad gjorde att det fungerade och vad var det som inte fungerade? Sammanställ gärna era observationer i någon form av tipslista eller handbok.
- Eventuellt kan de grupper som misslyckades få prova att skriva ny kod.

AVSNITT 4: PROGRAMMERINGSSPRÅK OCH BUGGAR

INNAN PROGRAMMET

Vilka olika teckenspråk kan eleverna i klassen? Vad har de språken gemensamt? Vad skiljer dem åt? Man kan teckna samma sak på olika sätt. På samma sätt fungerar programmeringsspråk. Det finns över 1000 olika programmeringsspråk, som används för olika ändamål. Alla programmeringsspråk har en väldigt bestämd grammatik och stavning. Gör man minsta lilla fel, fungerar ingenting.

UNDER PROGRAMMET

PAUSA! Allra sist får deltagarna ge var sitt exempel på vanliga fel när man skriver kod. Vanliga fel: *instruktioner i fel ordning, man har glömt ett moment i instruktionerna, man skriver vänster istället för höger.*

EFTER PROGRAMMET

1. Jämför programmeringsspråk

Använd Kopieringsunderlag 5 och titta på olika programmeringsspråk: JavaScript, C++ och Python. Vilka likheter har de? Vilka skillnader?

2. Hitta buggen

Använd Kopieringsunderlag 6 och låt eleverna arbeta parvis.

- En i varje par får sidan med den korrekta koden och den andra den kod som innehåller buggar. Utan att titta på varandras kod, ska de hitta felen.
- Gör liknande övningar där eleverna får samtala om text eller bild för att hitta skillnader. Använd klassiska finn-fem-fel-uppgifter från pysselböcker eller låt eleverna skapa egna exempel.

3. Hitta buggen i instruktioner (Tips! Titta på Programmera mera – Lektionstips)

Skriv ord, meningar eller instruktioner på tavlan som eleverna ska hitta felen i. Det kan vara olika typer av fel. Stavfel, grammatiska fel eller logiska fel, som att man inte kan börja skriva innan man hämtat en penna, eller gå ut genom en dörr innan man öppnat den. Glöm inte att påtala skillnaden mellan den mänskliga hjärnans förmåga att fylla i och läsa mellan raderna och datorns behov av exakta instruktioner.

4. Olika programmeringsspråk

Många tjänster som vi använder på datorn, är skrivna med flera olika språk.

Exempel

C++ Windows, Google Chrome
Python Instagram, Pinterest, Spotify, YouTube
JavaScript Många funktioner på en webbplats, som att fylla i formulär, klicka på en länk är programmerade i JavaScript. Hur texter och bilder ska se ut skrivs med HTML-kod. Färg och typsnitt på texten skrivs i CSS.

AVSNITT 5: BLOCKPROGRAMMERING OCH ALGORITMER

INNAN PROGRAMMET

Introducera begreppet algoritm. En algoritm är en instruktion som datorn ska följa. Algoritm går också att likna vid ett recept. I matematiken används begreppet algoritm för att beskriva uppställningar. Det är som ett recept för hur ett tal ska räknas ut. Villkor ligger bakom det mesta vi upplever i den digitala världen. Om vi gillar vissa filmer på en filmtjänst, kommer vi få förslag på liknande filmer. Kan eleverna komma på fler exempel på villkor som de möter? *Youtube, Netflix, Viaplay, Spotify, förslag på nya konton att följa på Instagram*.

UNDER PROGRAMMET

Deltagarna ska sätta ihop en algoritm för att sätta igång en film på datorn. Instruktionerna de använder är skrivna på magnetremsor som ska sättas upp på en tavla.
PAUSA! Kan eleverna se vilka remsor som ska vara med och i vilken ordning?

Den andra uppgiften handlar om att deltagarna skapa ett eget datorspel med hjälp av blockprogrammering. De har fått lite hjälp med vilka block de ska använda, men måste sätta allt i rätt ordning för att spelet ska fungera.
I sista uppgiften ska deltagarna programmera roboten så att den målar ett porträtt. För att koden ska få plats på bordet måste de förkorta den. Deltagarna programmerar roboten så att om det står måla, ska roboten använda funktionen som är en lista över vad den ska måla. En funktion är ett program i programmet som gör att man inte behöver skriva ut alla detaljer. Det här kan man fördjupa sig i om man vill, men det går också bra att hoppa över.

PAUSA! Som slutfråga vill programledaren få svar på vad en algoritm är.

EFTER PROGRAMMET

1. Algoritm för en saga version 1
Använd kopieringsunderlag 9.
- Klipp isär sagan i remsor. Låt eleverna pussla ihop sagan.
- Vilka ord avslöjar var i sagan de olika delarna hamnar?
- Plocka ut orden tillsammans och gör en generell algoritm för en saga.
- Skriv egna sagor efter algoritmen.

2. Vardagsalgoritm
Låt eleverna skriva egna algoritmer för vardagliga saker de gör. Kom ihåg vikten av att inte hoppa över något steg och att allt måste komma i rätt ordning.

3. Prova blockprogrammering
I det här avsnittet av Programmera mera använde deltagarna Scratch. Det finns flera olika lärarhandledningar och guider till hur Scratch fungerar. För att se just det spel som deltagarna skapade går det att söka på ”programmera-mera” på sidan scratch.mit.edu. Det fungerar bara på dator och inte på surfplatta eller mobil.
Andra resurser för blockprogrammering är code.org och madewithcode, som fungerar i alla webbläsare och appar som Hopscotch, Pyonkee, Tynker och ScratchJR.

4. Spela Villkorsspelet
Använd Kopieringsunderlag 10 till Villkorsspelet och spela tillsammans. Spelet tränar att läsa kod med villkorssatser, alltså if, else och else if.

Material
- kortlek
- penna och papper

Spela
- Låt kortleken ligga mitt på bordet.
- Turas om att vända upp ett kort.
- Sätt poäng efter algoritmen. Vinnare är den som har flest poäng efter 5 omgångar.

Algoritm
if (kort lägre än 5)
if (kort svart)
then – Samma antal poäng till dig, som valören på kortet
else – 1 poäng till de andra
else if (kort hjärter) 1 poäng till dig

5. Bygg ett eget datorspel i klassrummet.
- Använd A4-papper i olika färger som spelplan.
- Lägg ut en bana mellan bord och stolar.
- Anpassa svårighetsgraden för gruppen.
- Det går utmärkt att programmera någon till att vara zombie som går i en loop, eller liknande varianter.
- Bestäm tillsammans vad de olika färgerna ska betyda. Skriv upp villkoren.

Exempel
- OM gul DÅ plocka upp
- OM röd DÅ hoppa över

- Arbeta i par
- Den ena personen ska programmera, den andra ska ta sig genom en bana. Det enklaste är att programmeraren skriver ner kod till spelaren som tar sig runt. Spelaren måste följa koden, även om det blir fel.
- En i taget provar koden. Då ser alla hur de andra programmerarna har tänkt.
- När eleverna har fattat hur man gör, kan de konstruera nya spel. Skolgården, fritids eller lektionerna i idrott och hälsa är utmärkta tillfällen för att jobba vidare med den här övningen.

NÄR NI HAR SETT ALLA AVSNITT

Prata om olika förmågor, färdigheter och egenskaper som är bra att ha om man ska bli duktig på att programmera. Tålamod, noggrann, analytisk, problemlösare, gilla att sortera saker, gilla att se hur saker hänger ihop, bra på att se mönster och när saker inte stämmer, tänka logiskt, hålla reda på många saker samtidigt, kunna hitta på nya lösningar, våga prova nya sätt.

FÖR DIG SOM VILL GÖRA MER

Att programmera handlar ofta om att läsa kod som andra har skrivit, eller om att läsa sin egen när det har blivit fel. Det är en ny slags läsning; att lära sig läsa kod. Det gäller att se framför sig i huvudet vad koden betyder. Som med all läsning, är det bara att träna för att lära sig och bli bättre. Här följer några uppgifter som kan vara roliga att göra efter att ni har sett alla avsnitt i serien.

1. Para ihop kod och bild

Använd Kopieringsunderlag 11 med kod och bild

- Dela ut underlaget individuellt eller parvis.
- Titta på bilderna och kodsnuttarna från programmet. Hur kan man lösa uppgiften, även om man inte kan någonting om just det här programmeringsspråket?
- Börja med att titta på vad som är exakt lika i alla kodsnuttar. Sudda, sakta(10) och osynlig finns i alla kodsnuttar och kan därför strykas. Det är kommandon som instruerar datorn hur den ska hantera själva programmet och har inte med bilden att göra.
- Alla kodsnuttar innehåller cirkel(x) men de måste så klart vara kvar, eftersom de är grunden i figuren som ritats ut.
- Det finns upprepa(4) och upprepa(10) –vilka bilder kan de höra till?
- Det finns två olika kommandon med färger: färg(gul) och fyll(röd) och tvärtom. Vad kan det betyda? Titta på bilderna och dela upp dem efter vilka som troligtvis är skrivna med samma kod.
- Det knepigaste med bilderna är att det finns en kvadrat mellan cirklarna i figur x och y. Det går inte att hitta någon kvadrat direkt i koden. Om man är klurig så listar man ut att upprepa(4) kan ha något med en kvadrat att göra.

2. Figurjakt i klassrummet

Använd Kopieringsunderlag 12 och 13.

- Eleverna ska ha varsin lapp med kod.
- Underlaget med bilder kan kopieras 1-3 gånger, beroende på hur stor gruppen är.
- Klipp ut alla bilder och placera ut bilderna på olika ställen i klassrummet.
- Dela ut en lapp med kod till varje elev. (Det finns olika svårighetsgrader så dela så att varje elev får en uppgift som passar dem.)
- Se till att alla förstår vad koden betyder.
- Berätta att det finns geometriska figurer på olika platser i klassrummet.
- Utan att prata med varandra, går eleverna runt och letar efter den figur som passar till koden på deras lapp. Det finns tre olika sätt att skriva kod till varje figur.
- När alla har hittat den figur de tror är den rätta kan eleverna jämföra med varandra och se om det verkar som att de har hamnat rätt.

NÅGRA HISTORISKA TILLBAKABLICKAR

EN FÖRSTA BÖRJAN

Under 1830- och 40-talet jobbade matematikern Charles Babbage på konstruktionen av en stor och avancerad beräkningsmaskin. Den kallades ”den analytiska maskinen”. Ada Lovelace, som också var matematiker, lärde känna Charles och de började samarbeta. Ada skrev bland annat en text som förklarade den analytiska maskinen. I sina noter till den texten skrev Ada Lovelace vad som kom att bli världens första algoritm.

Den analytiska maskinen byggdes aldrig färdigt. Men Ada var troligen först i världen med att förstå vilka möjligheter som fanns i att bygga en maskin och programmera den.

Den andra tisdagen i oktober varje år kallas för Ada Lovelace-dagen. Det är en internationell temadag för att ära Ada och andra kvinnor inom vetenskap, teknologi, ingenjörskap och matematik. Det finns även ett programmeringsspråk uppkallat efter Ada. Många kallar Ada Lovelace för världens första programmerare.

DATORER

Under andra världskriget blev det viktigt att kunna bearbeta väldigt stora mängder information, eller data. Bland annat för att kunna knäcka hemliga koder som fienden använde.

Den brittiska matematikern Alan Turing var en pionjär inom datavetenskapen. Han brukar ibland kallas för ”*the father of computing*” (ungefär ”pappa till datavetenskap). Under andra världskriget arbetade han vid *Bletchley Park* som var högkvarter för Storbritanniens kod- och chifferbyrås (GCCS). Turing är bland annat känd för att han knäckte det tyska chiffersystemet Enigma. Det var också vid Bletchley Park som den första programmerbara datorn, Colossus, byggdes. Den knäckte ett annat chiffer, Lorentz.

Nästan samtidigt byggdes en dator i USA, ENIAC. Alla 6 huvudprogrammerare till ENIAC var kvinnor. De hette Kay McNulty, Betty Jennings, Betty Snyder, Marlyn Wescoff, Fran Bilas and Ruth Lichterman. En annan tidig dator var CSIRAC i Australien.

I Sverige byggdes BARK och BESK på 1950-talet av Matematikmaskinnämnden. Under några veckor 1953 var BESK den snabbaste datorn i världen. BESK var så stor att den fyllde ett helt rum på Tekniska högskolan.

ROBOTAR

En robot är en maskin som styrs av programmering för att den ska kunna utföra fysiska uppgifter. Själva ordet robot har funnits länge på flera språk och betyder ungefär ”slitigt arbete”. Den första industriroboten installerades i en bilfabrik år 1961. Den hette Unimate. Unimate var en enkel robot vars enda uppgift var att. plocka komponenter från ett löpande band och svetsa fast dem på bilarna. Det var arbetsuppgifter som var riskfyllda för dem som jobbade i fabriken.

En robot har ett styrsystem som skickar signaler om vad roboten ska utföra och det är programmering som bestämmer vad det är den ska göra. Så en robot kan alltså omprogrammeras för att göra något annat. En robot kan se ut nästan hur som helst.

Leonardo da Vinci ritade faktiskt något som liknade en robot redan på 1400-talet. Det var en sorts ”robotriddare”. Men Leonardo da Vinci kallade den inte för robot.

ETTOR OCH NOLLOR

En dator kan bara förstå två saker: ETT och NOLL. Eller, egentligen är det inte en etta och en nolla heller, det är PÅ eller AV. Man kan likna det vid en lampa som tänds och släcks. En etta betyder tänd och en nolla betyder släck. Det är allt en dator behöver veta för att kunna göra hur mycket som helst.

Raderna av miljontals ettor och nollor kallas för binärkod, eller maskinkod. Maskinkod är svårt att läsa för en människa. Därför har vi skapat olika programmeringsspråk. Inne i datorn sitter en kompilator. Kompilatorn översätter programmeringsspråk till maskinkod.

Grace Hopper var matematiker och amiral i amerikanska flottan (Navy Admiral). På femtiotalet skapade hon den första kompilatorn som kunde omvandla ett programmeringsspråk till maskinkod. Hon var också med och utvecklade flera programmeringsspråk.

GPS: GLOBAL POSITIONING SYSTEM

Bakgrunden till dagens GPS är att teamet som skötte satelliten Sputnik märke att radiosignalerna som Sputnik sände ut ändrades när satelliten kom närmare eller åkte längre bort. Genom att hålla koll på signalerna gick det att hålla koll på satelliten och åt vilket håll den rörde sig.

Det första satellitbaserade navigationssystemet hette TRANSIT och byggdes 1959.

GPS fungerar med hjälp av triangulering. Satelliterna berättar hela tiden var de är och en satellit som hör din signal kan räkna ut hur långt bort du är. Men du kan fortfarande vara var som helst runt om satelliten. Med hjälp av minst två satelliter till går det att bestämma mer exakt var du befinner sig när du sänder din signal.

DATORSPEL

De första datorspelen var väldigt enkla, men ändå spännande och nya för sin tid. Ett av de första spelen hette *Tennis for Two*. Det byggdes av William Higinbotham till *The Brookhaven National Laboratory's* årliga utställning. Han ville använda spelet för att göra utställningen mindre statisk. Och han lyckades! Hundratals besökare köade för att få spela Tennis for two.

På 1970-talet dök det upp datorspel i spelautomater på caféer. När hemdatorn kom så blev det också möjligt att ha dataspel, och konsolspel, hemma. Mario Bros, Pac Man och Donkey Kong är exempel på spel från tidigt 80-tal.

ARPANET OCH INTERNET

Idén bakom internet var att forskare ville koppla ihop sina stora datorer för att kunna dela information. En forskargrupp som hette ARPA (*Advanced Research Project Agency*) blev de första som lyckades koppla samman sina datorer i ett nätverk. Nätverket kallades för ARPANET.

Första gången som två datorer fick kontakt var kvällen den 29 oktober 1969. Då lyckades Charley Kline på UCLA (*University of California, Los Angeles*) logga in på Bill Duvalls dator på *Stanford University* (nära *San Fransciscо*). Planen var att skriva ”login”. Bill kunde knappt tro sina ögon när han såg Charleys ”l” och ”o” dyka upp på sin skärm. Tyvärr kraschade Bills dator efter två bokstäver. Så det allra första meddelandet som skickades mellan två datorer blev bara ”lo”. Men lite senare samma kväll lyckades Charley skriva hela ”login” och logga in på Bills dator. Från början var det bara 4 datorer i USA som var uppkopplade, men för varje år hakade fler och fler på. År 1973 kom Norge och Storbritannien med och Sverige anslöt sig tio år senare, 1983. Det var samma år som man lämnade ARPANET och gick över till det nya systemet med IP-adresser och TCP, det vi idag kallar internet.

I början använde man telefonjacket för att ansluta till internet. Då kunde man inte prata i telefon och använda internet samtidigt. Men sedan kom bredband och fiber och nu går det mycket fortare att skicka data.

MUSIK

Toner kan bildas på många olika sätt. På en gitarr är det strängens längd och tjocklek som avgör. På en flöjt skapas olika toner genom att luften träffar en kant på olika sätt. I en synth (syntheziser) är det möjligt att skapa ljud med hjälp av dataprogram.

Redan Ada Lovelace funderade på möjligheten att skapa musik med maskiner. År 1957 uppfann elektroingenjören Max Mathews dataprogrammet MUSIC. Med det kunde en dator spela en 17 sekunder lång komposition. Under hela sitt liv fortsatte Max att utforska och experimentera med hur en dator går att använda för att skapa musik. Max Mathews kallas ibland ”pappan till datamusik” (*father of computer music*).

RYMDEN

Hur kan det komma signaler från ett ställe där ingen människa eller maskin varit förr? Jo, de skickas från rymdsonden Voyager 1. Den befinner sig nu utanför vårt solsystem, i rymden mellan stjärnorna, i den interstellära rymden. Voyagerprojektet består av de båda obemannade rymdsonderna Voyager 1 och Voyager 2. Båda farkosterna sändes ut 1977 för att undersöka planeter i vårt solsystem. Sedan dess har de levererat bilder och unika mätningar från de yttre planeterna Saturnus, Jupiter, Uranus och Neptunus.

Den 14 februari 1990, efter att Voyager 1 hade passerat både Neptunus och Plutos omloppsbana, vände NASA (*National Aeronautics and Space Administration*) på rymdsonden så att kameran pekade tillbaka mot solen. På så sätt kunde Voyager 1 ta en serie familjeporträtt av planeterna i solsystemet. Det var också vid det tillfället som det berömda fotot ”*Pale Blue Dot*” (en blek blå prick) togs av jorden. Därefter stängdes kameran av för att spara energi och minne.

Förr eller senare kommer vi att tappa kontakten med Voyager 1. Antingen går den sönder eller kommer så långt bort att signalerna inte längre når Jorden.

ARTIFICIELL INTELLIGENS

Om en dator bara gör det som den är programmerad till att göra, kan man då programmera den till att tänka själv? Idag är det många forskare och företag som försöker. Det kallas Artificiell Intelligens, AI. På senare tid har de gjort stora framsteg. Till exempel när det gäller att få datorn att förstå bilder.

En som tidigt funderade på artificiell intelligens var Alan Turing. Han ställde sig frågan: ”Om en maskin kunde tänka, hur skulle vi veta det?” Baserat på den frågan utvecklade han Turingtestet. Om man kan programmera en dator så att den kan svara på frågor utan att man vet om det är en människa eller en dator som svarar - då har den klarat Turingtestet.

TIDSLINJE

1840-tal:	Ada Lovelace skriver världens första algoritm i noterna till sin text om den analytiska maskinen som Charles Babbage arbetade med
1940-tal:	Första datorerna kommer
1950-tal:	Första svenska datorer, BARK och BESK och SMIL. Sputnik 1, tidiga datorspel (Tea for Two) Turingtestet
1960-tal:	Industri-robotar, ARPANET
1970-tal:	Arkadspel, Voyager 1 lämnar jorden
1980-tal:	Dator i hemmen
1990-tal:	Internet i hemmen
2000-tal:	Wifi hemma, Touch-screen i hemmen
2010-tal:	Smarta telefoner blir supervanliga, Hour of Code, programmering i skolan? Små robotar i hemmen.
2020-tal:	Vad händer när du blir vuxen?

TACK TILL

Utmaningarna i programmet och övningarna i handledningen är framtagna i samråd med professor Peter Parnes, docent Fredrik Heintz och Ellen Sundh. Programmeringstävlingen bebras.se. har fungerat som inspiration och för den pedagogiska expertisen står Karin Nygårds som har utformat övningarna utifrån sin egen fleråriga undervisning inom programmering.

TIPS TILL DIG SOM VILL TITTA, LYSSNA ELLER LÄSA MER

FLER PROGRAM FRÅN UR:

Robotdagen 2015 Vems är skulden när en robot gör fel, UR-Samtiden
From Business to Buttons 2016-Robortkärlek, UR Samtiden
Tema-Roboten tar plats, UR Samtiden

FÖR BARN OCH UNGA

Bygg ditt eget dataspel! Måns Jonasson.
Koda bus och dataspel! Måns Jonasson.
Programmera med Scratch Jr, Marina Umaschi Bers och Mitchel Resnick.
Curly Bracket - den gömda koden och Curly Bracket - Corpuratus hemlighet, Johan Wendt och Tor Moström. Finns även en tillhörande webbsida.
Hej Ruby, Linda Liukas. Finns även tillhörande webbsida och lärarhandledning.
Så funkar Internet, Karin Nygårds.

"Kodboken" En gratistjänst från Kodcentrum. Länktips på böcker och annan läsning som handlar om programmering och digitalt skapande, datalogiskt tänkande och digital kompetens.

FÖR VUXNA

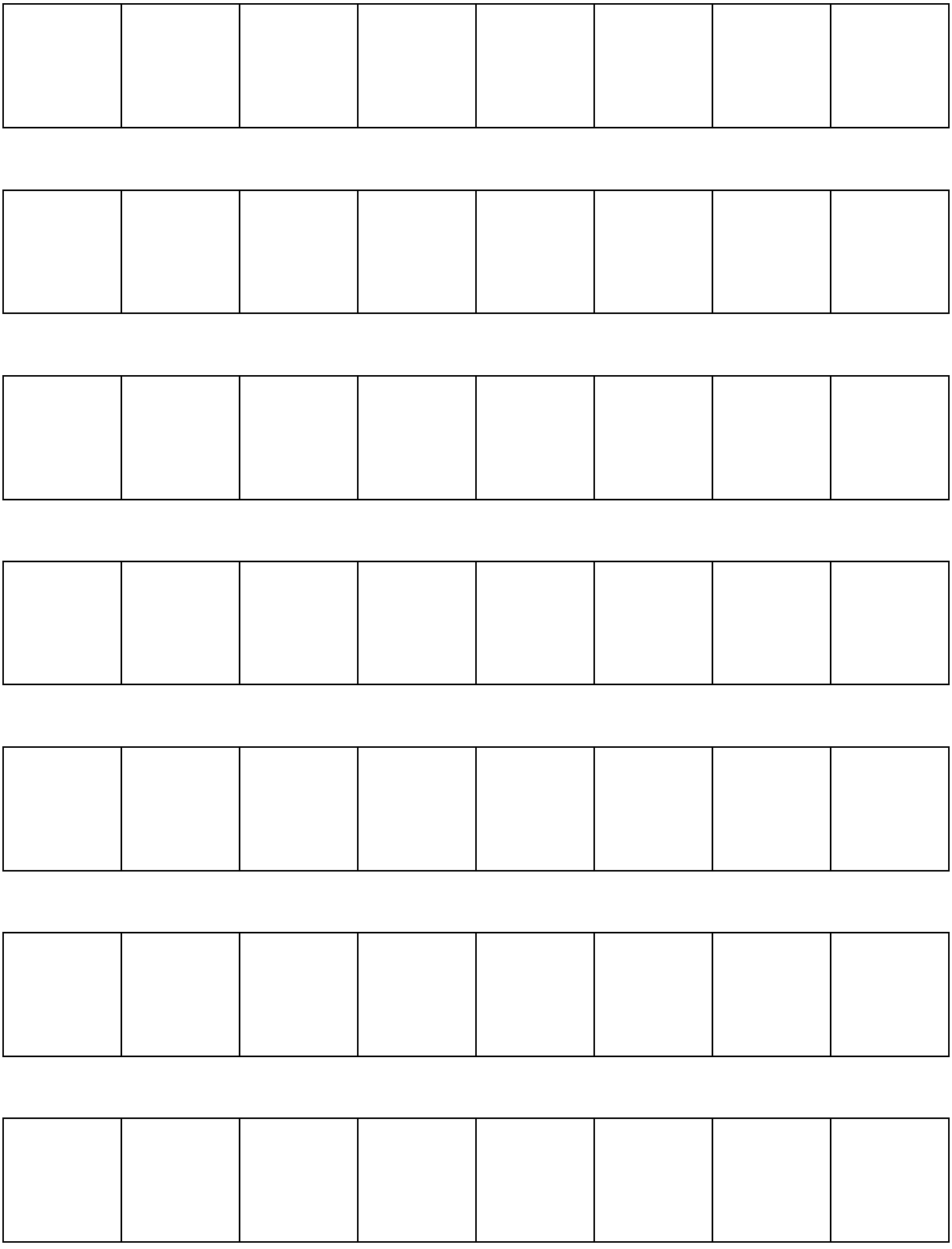
Koden till digital kompetens, Karin Nygårds
Så funkar Internet, Karin Nygårds
Hjälp ditt barn med programmering, Carol Vorderman
Starta med programmering, Erika Olsson
Programmering för högstadiet, Mikael Tylmad och Pontus Walck
Program or be programmed, Dr Douglas Rushkoff
Hjälp ditt barn med programmering : en illustrerad guide som lär...
Carol Vorderman

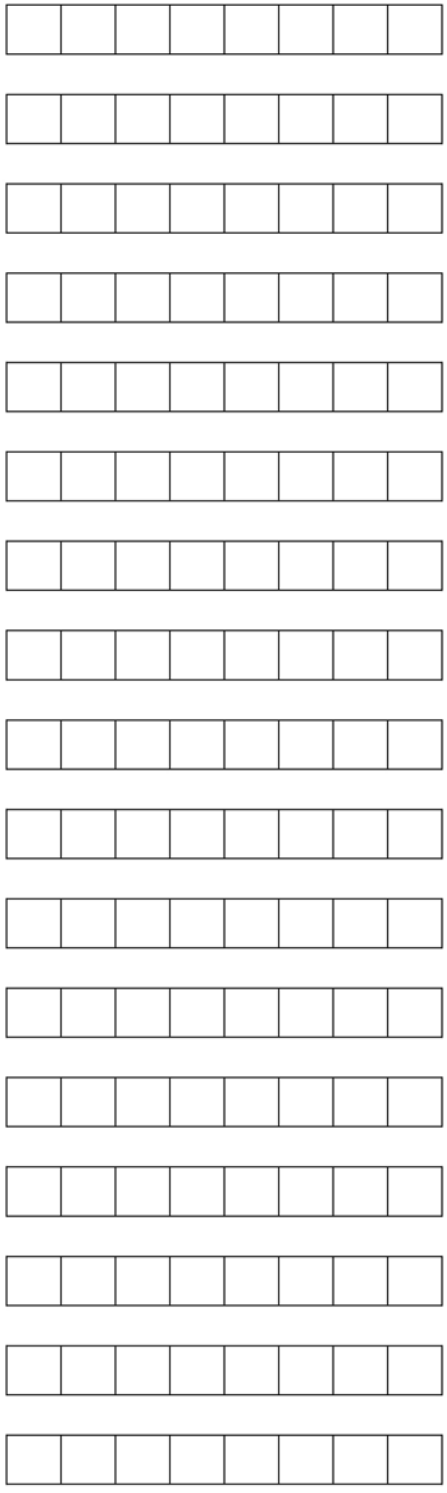
HANDLEDNINGAR FÖR LÄRARE OCH PEDAGOGER

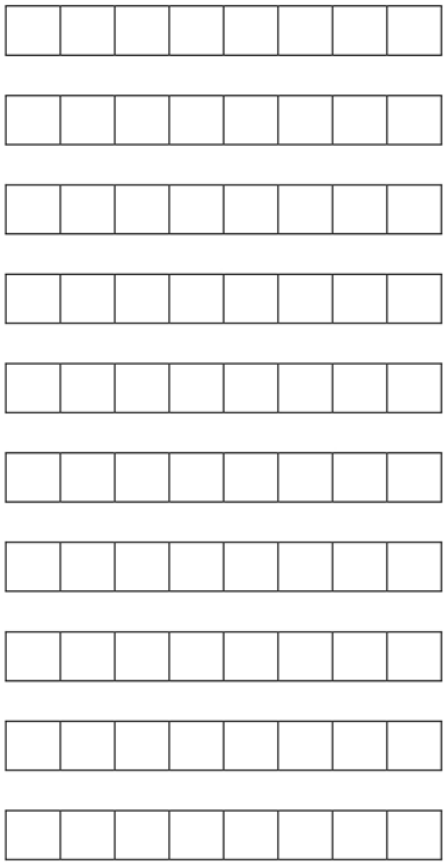
Kom igång med Scratch, från Internetstiftelsen i Sverige (IIS)
Barnhack!, från Internetstiftelsen i Sverige (IIS)
Nyckeln till Hej Ruby, Linda Liukas och Volante Förlag (pdf att ladda ned)

0100 0001	A
0100 0010	B
0100 0011	C
0100 0100	D
0100 0101	E
0100 0110	F
0100 0111	G
0100 1000	H
0100 1001	I
0100 1010	J
0100 1011	K
0100 1100	L
0100 1101	M
0100 1110	N
0100 1111	O
0101 0000	P
0101 0001	Q
0101 0010	R
0101 0011	S
0101 0100	T
0101 0101	U
0101 0110	V
0101 0111	W
0101 1000	X
0101 1001	Y
0101 1010	Z

Skriv med maskinkod

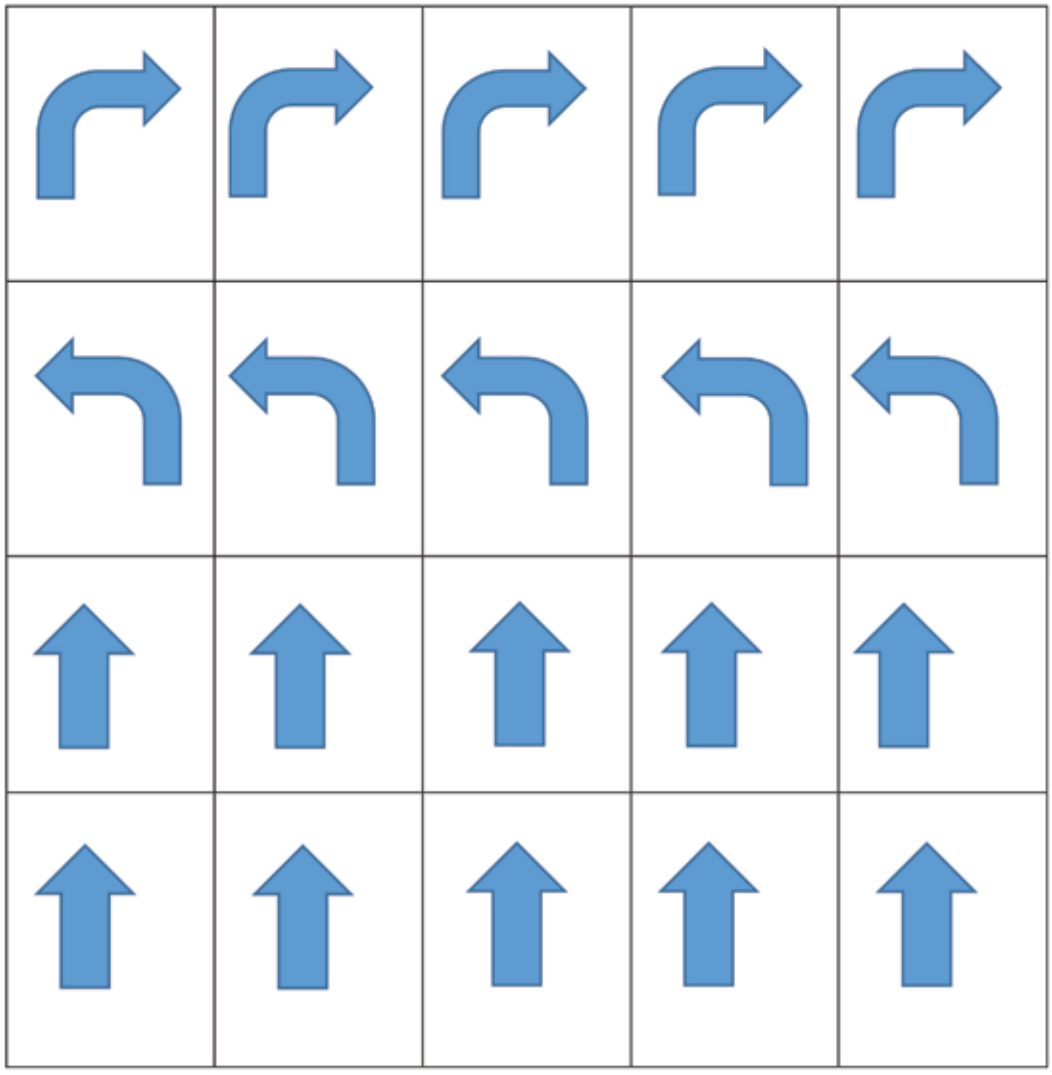






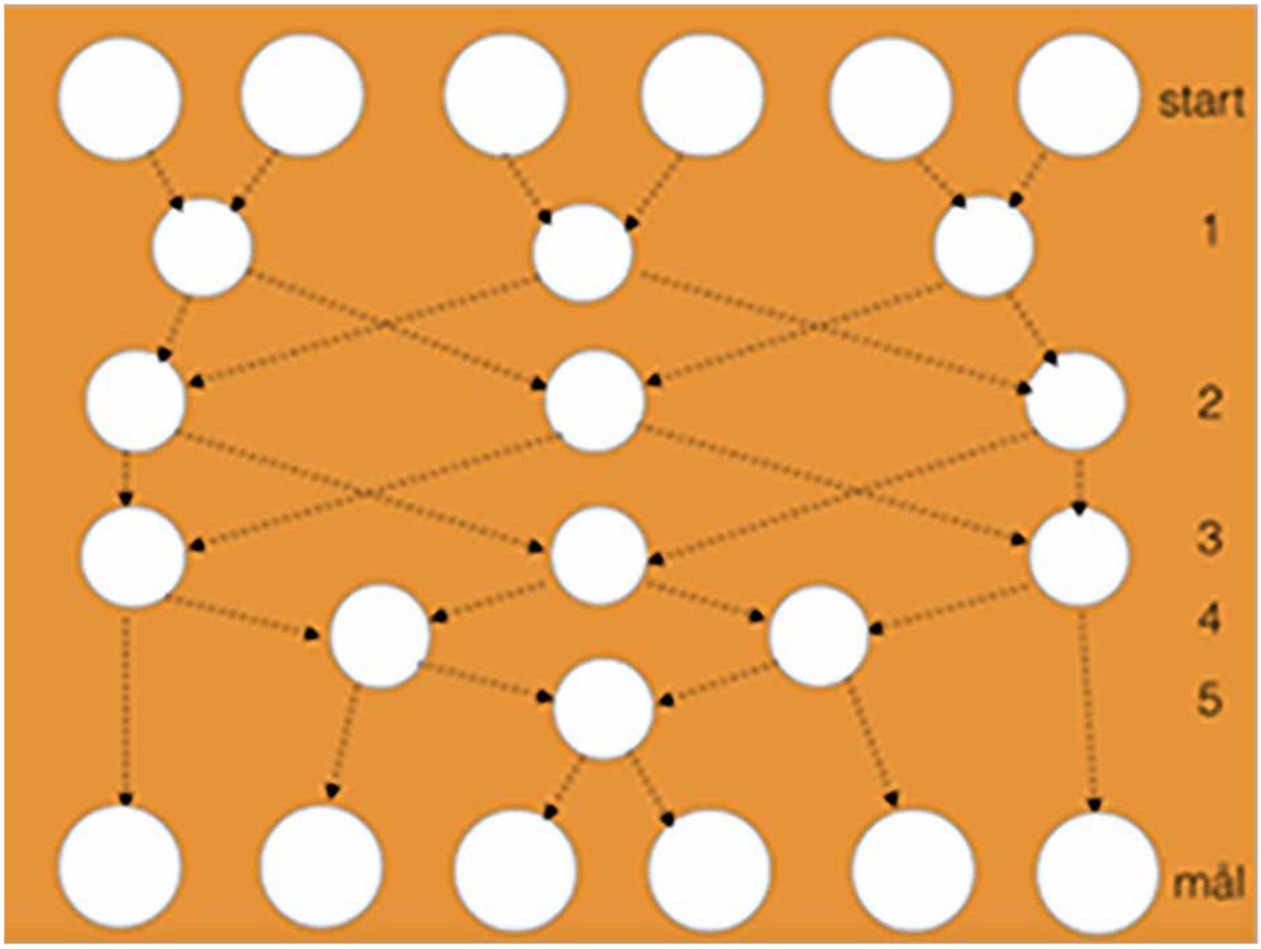
PILPROGRAMMERING

1. Lägg ut start och mål
2. Lägg ut hinder om ni vill göra det svårare
3. Lägg ut pilarna i den ordning som figuren ska gå
4. Prova koden



Klipp ut pilarna.
Svängpilarna betyder bara att figuren vrider sig, inte att den tar ett steg åt höger eller vänster. För att komma åt höger eller vänster måste svängen följas av en pil fram.

KOPIERINGSUNDERLAG 4



Jämför programmeringsspråk

Så här ser koden ut för att få datorn att skriva ut ”Programmera mera!”, på tre olika språk.

Javascript

```
alert(”Programmera mera!”);
```

Python

```
print ”Programmera mera!\n”
```

C++

```
#include <iostream>
int main()
{
	std::cout << ”Programmera mera!” <<std::endl;
}
```

Vilka tecken återkommer?

Hur skulle koden ändras om du vill att datorn ska skriva ut ditt namn?

Hitta buggen (rätt kod)

Javascript

```
alert(”Programmera mera!”);
```

Python

```
print ”Programmera mera!\n”
```

C++

```
#include <iostream>
int main()
{
	std::cout << ”Programmera mera!” <<std::endl;
}
```

Hitta buggen (fel kod)

Javascript

```
alert(”Programmera mera!):
```

Python

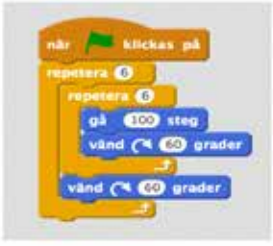
```
Print( ”Programmera mera!\\n”)
```

C++

```
#include <iostream>
int main()
	std::cout < ”Programmera mera!” <<std:endl;
}
```

KOPIERINGSUNDERLAG 7

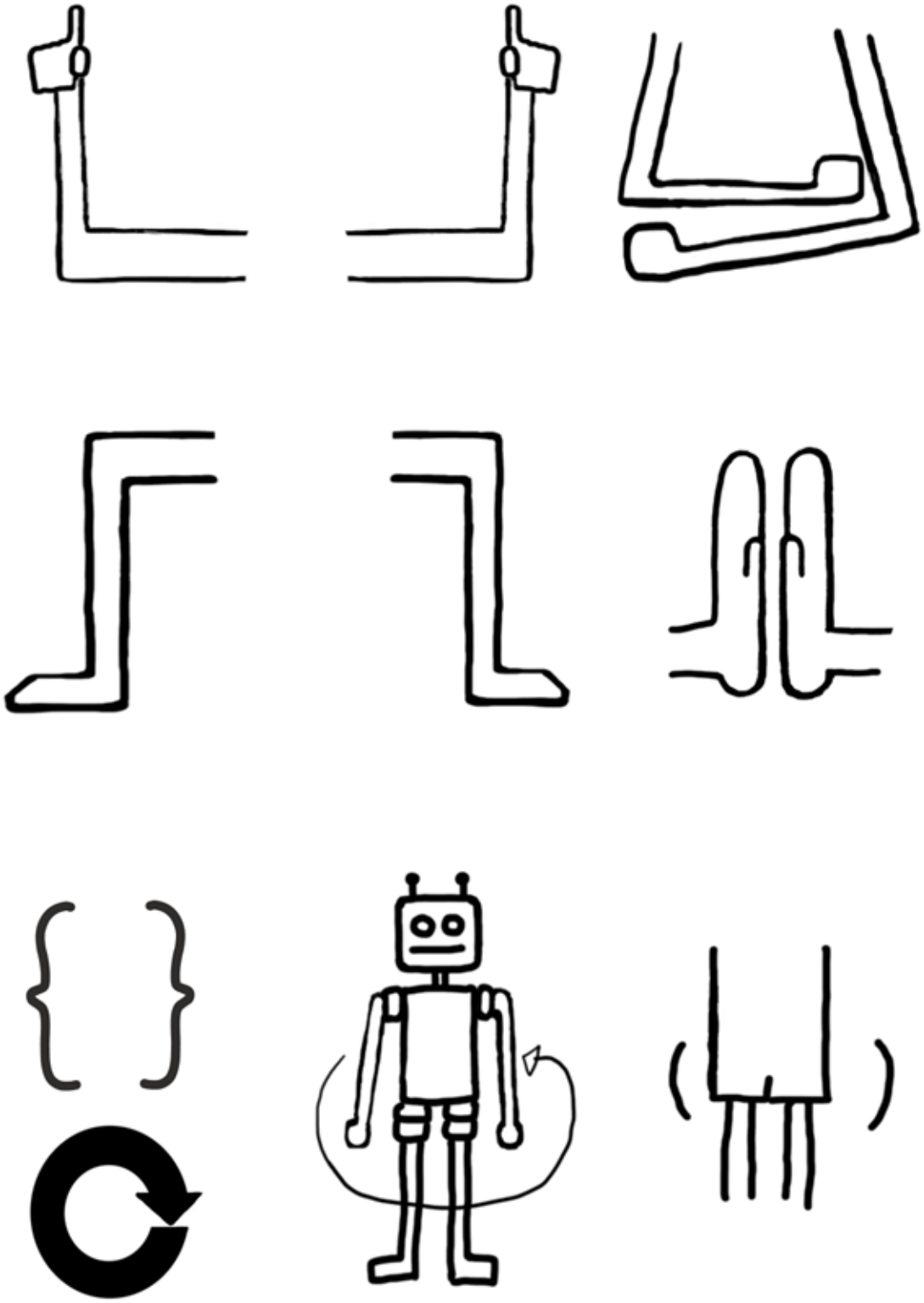

```
upprepa(6)
{fram(100)
höger(60)}
```




```
upprepa(6){
upprepa(6)
{fram(100)
höger(60)}
höger(60)}
```


```
upprepa(3)
{fram(100)
höger(120)}
```

KOPIERINGSUNDERLAG 8



KOPIERINGSUNDERLAG 9

Det var en gång ett stort och läskigt monster. Det bodde i en grotta dit inga människor vågade gå. Monstret kände sig väldigt ensam och ledsen.

Plötsligt en dag kom en liten flicka till grottan. Hon hade gått vilse och letade efter någonstans att sova. Monstret blev så glad över att få sällskap och flickan blev glad över att hon hittade någonstans att sova.

Nästa dag kunde monstret visa flickan var hennes by låg, så att hon kunde hitta hem. Flickan ville visa monstret var hon bodde.

Men när monstret och flickan visade sig i byn blev alla förskräckligt rädda och flydde åt alla håll. Jägaren i byn hämtade sitt gevär för att skjuta monstret.

Då ställde sig flickan i vägen. Hon förklarade att monstret var den som hade hjälpt henne hem och att det var ett snällt monster.

Till slut vågade alla bybor komma fram och flickans familj hälsade på monstret och tackade för hjälpen.

Sen levde de lyckliga i alla sina dagar.

KOPIERINGSUNDERLAG 10

Skriv saga efter algoritm 2

1. en/ett person/djur/väsen
2. en/ett person/djur/väsen
3. en plats, byggnad
4. ett verb
5. ett adverb
6. ett substantiv
7. ett verb
8. ett verb
9. ett adjektiv
10. ett adjektiv

Det var en gång 1_____ och 2_____
som bodde i 3_____.
Varje dag brukade (1&2) 4_____
En dag hände något 5_____.
Det var en stor 6 _____
som plötsligt hade 7_____.
Som tur var kunde (1)8_____.
Det tyckte (2)var mycket 9_____.
Sen levde de 10_____ i alla sina dagar.

Villkorsspelet
Det här spelet tränar att läsa kod med villkorssatser, alltså if, else och else if

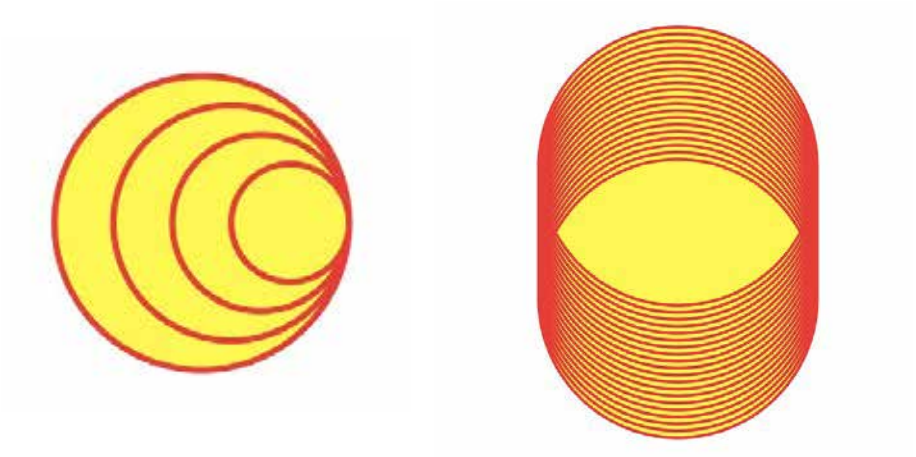
Material som behövs:
Kortlek
Penna och papper

Låt kortleken ligga mitt på bordet. Turas om att vända upp ett kort. Sätt poäng efter algoritmen nedan.
Vinnare är den som har flest poäng efter 5 omgångar.

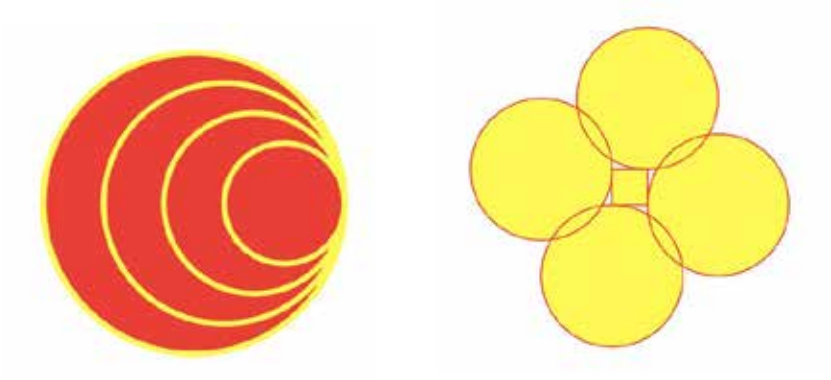
if (kort lägre än 5)
if (kort svart)
then – Samma antal poäng till dig, som valören på kortet

else – I poäng till de andra
else if (kort hjärter) I poäng till dig

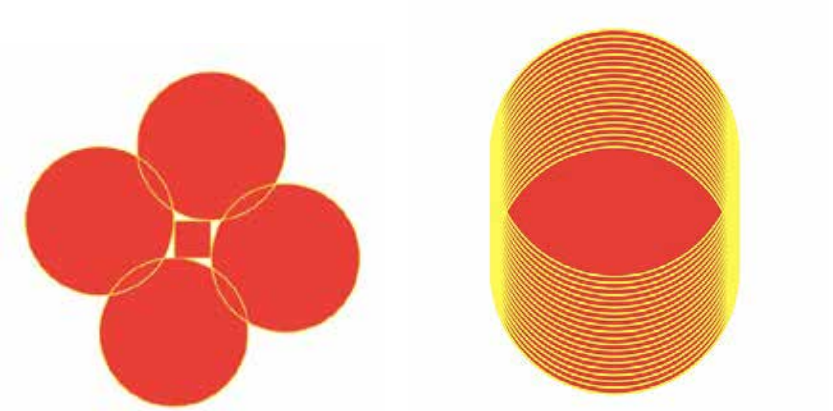
1. sudda
2. sakta(10)
3. färg(gul)
4. fyll(röd)
5. upprepa(10) {
6. cirkel(100)
7. fram(5)
8. cirkel(100)
9. fram(5)
10. }



1. sudda
2. sakta(10)
3. färg(röd)
4. fyll(gul)
5. upprepa(4) {
6. fram(5)
7. cirkel(100)
8. höger
9. }



1. sudda
2. sakta(10)
3. färg(gul)
4. fyll(röd)
5. cirkel(50)
6. cirkel(40)
7. cirkel(30)
8. cirkel(20)



KOPIERINGSUNDERLAG I3