UNDERVISNINGS
MINISTERIET
STYRELSEN FOR
UNDERVISNING OG KVALITET

Programmering C, valgfag

Vejledning

Undervisningsministeriet
Styrelsen for Undervisning og Kvalitet
Gymnasiekontoret, marts 2018

Vejledningen præciserer, kommenterer, uddyber og giver anbefalinger vedrørende udvalgte dele af læreplanens tekst, men indfører ikke nye bindende krav.

Citater fra læreplanen er anført i kursiv.

Indholdsfortegnelse

1. Identitet og formål

1.1. Identitet

1.2. Formål

Programmering er et fagfelt der kan udvikle elevernes innovative og skabende kompetencer. Det handler ikke bare om, at eleverne skal kunne programmere. Det handler også om at kunne tænke selv, kunne samarbejde omkring idéudvikling, skabelse og videndeling. Hermed er programmering også med til at udvikle elevernes digitale- og studiekompetencer. Programmering kan medvirke til at skabe interesse for teknologi og it, nationalt og globalt, og dermed også bidrage til elevernes karrierekompetencer, samtidig med at de bliver introduceret til abstrakt tænkning, logisk opbygning og problemløsning.

Blandt de faglige mål indgår, at eleverne skal kunne:

- *demonstrere viden om fagets identitet og metoder,*
- *løse en enkelt problemstilling gennem udviklingen af et program bl.a. i samspil med andre fag.*

Dette betyder, at der ikke alene undervises i programmering, men også om programmering. Når faget indgår i et samarbejde med andre fag, uanset om det er i studieområdet (SO), studieområdeprojektet (SOP) på hhx/htx eller studieretningsopgaven (SRO), studieretningsprojektet (SRP) på stx, større skriftlige opgave (SSO) på hf eller det er et fagligt samarbejde i en studieretning, så skal eleverne kunne inddrage og anvende relevante programmeringsmæssige metoder, redegøre for disse i et sprog, som man også uden for faget kan forstå, samt forholde sig til fagets muligheder og begrænsninger i arbejdet med den konkrete problemstilling.

I læreplanens afsnit 1.1 og 1.2 er der givet en kompakt beskrivelse af fagets identitet og af det overordnede formål med undervisningen.

2. Faglige mål og fagligt indhold

2.1. Faglige mål

Programmering i gymnasiet giver eleverne muligheder for selv at skabe produkter fra bunden. Ved at lære om programmers opbygning og mulighederne i et programmeringssprog eller en programmeringsplatform, kan eleverne indgå i en kreativ udforskning af et fag- eller problemområde, og derigennem skabe værdi eller ny viden for dem selv eller andre.

Eleverne skal kunne:

- *læse enkle programmer og redegøre for deres funktionsmåde og anvendelsesmuligheder*
- *rette og tilpasse enkle programmer*

Elevernes forudsætninger for at kunne komme i gang med programmering varierer kraftigt. Mange elever kender i et eller andet omfang til programmering - eller er måske endda meget erfarne - mens andre har meget begrænset forhåndskendskab. De faglige mål giver plads til hele spektret af elever, men underviseren bør tilrettelægge forløb og aktiviteter, så nybegynderne får plads og tid til at lære om syntaks, datatyper, kontrolstrukturer, osv., mens de dygtigere elever udfordres på andre områder.

anvende eksisterende programdele og biblioteksmoduler i arbejdet med at programmere et fungerende system

demonstrere systematik i programmeringsprocessen

Gode arbejdsvaner er også i programmering en afgørende hjælp til at holde styr på og bevare overblikket. Undervisningen skal derfor give eleverne redskaber, der kan støtte dem i en målrettet og systematisk arbejdsproces hen imod løsningen af en given problemstilling. Her vil abstrakte dokumentationsformer som pseudokode, flowcharts, og lignende modeller kunne støtte eleverne i udviklingen af dele af koden

– *løse en enkel problemstilling gennem udviklingen af et program bl.a. i samspil med andre fag*

Det er oplagt at tage udgangspunkt i problemstillinger fra elevernes hverdag; problemstillinger fra andre fag, programmer der ligner dem eleverne bruger til hverdag på forskellige platforme, eller simpelthen det, der interesserer den enkelte elev. Mulighederne for at bruge programmering til at skabe programmer der på en eller anden måde kan bruges i elevernes hverdag udvides til stadighed, med apps, programmerbare komponenter, Internet of Things, og den hurtigt voksende mængde data, der er tilgængelig på nettet. Formålet er ikke at eleverne skal lære en masse forskellige programmeringssprog eller teknologier at kende, men at de indser, at de kompetencer de tilegner sig kan anvendes i en bred vifte af aktiviteter. Det gøres ved at lade eleverne være kreative og udforskende i deres arbejde med udviklingen af programmer. Behandling af forskellige problemstillinger fra andre fag vil styrke elevernes karrierekompetence, idet de opnår forståelse for hvordan programmering som professionelt fag sjældent arbejder på at løse fagets egne problemstillinger, men ofte på at bidrage til løsning af problemstillinger fra andre fagområder. Inddragelse af programmering som et værktøj i andre fag vil ligeledes styrke elevernes muligheder for innovation.

I slutningen af forløbet skal eleven selv kunne løse en enkel problemstilling gennem udviklingen af et program. Her skal indgå hensigtsmæssige arbejdsgange og abstrakte beskrivelser af programmeringen. Som underviser skal man hjælpe eleverne med at finde en passende problemstilling, der både er realistisk for dem at udforme en (som minimum delvis) løsning til, og som lægger op til en systematisk, trinvis udviklingsproces I forbindelse med vejledningen af eleven, bør man prøve at sikre sig, at eksamensprojektet formuleres på en måde, så eleven får mulighed for at anvende et bredt udvalg af de elementer, som man har arbejdet med i timerne.

– *anvende grundlæggende konstruktioner i et programmeringssprog*
– *demonstrere viden om fagets identitet og metoder*

Programmeringsfaget adskiller sig fra de andre fag i gymnasiet, ved at være det fag hvor den iterative udviklingsmetode for alvor giver mening. Eleverne ved allerede - fra programmer de selv bruger på deres computer, telefon, osv. - at programmer altid er under udvikling, og kun sjældent anses for at være færdige produkter. Dette kan eleverne overføre til deres eget arbejde, og bevidstheden om at programmeringen ikke har til formål at gøre programmet færdigt, men at tage et lille skridt ad gangen mod en løsning, kan hjælpe alle elever med at opnå tilfredsstillende resultater i faget

2.2. Kernestof

– *- programmeringssprog og elementer i programmeringssprogets opbygning såsom data- og kontrolstrukturer*

Det er tilrådeligt, at der kun vælges ét primært programmeringssprog af hensyn til den pædagogiske tilrettelæggelse og overskueligheden for eleverne. I en differentieret undervisningssituation kan der vælges et sekundært sprog til nogle elever. I et tværfagligt samarbejde med eksempelvis matematik, kan matematikprogrammer såsom Maple og Mathcad godt fungere sideløbende med det valgte sprog.

Det valgte sprog bør give mulighed for at arbejde med grundlæggende data- og kontrolstrukturer så eleven får en god forståelse af programmeringssprogets opbygning.

Herunder nogle forslag til temaer inden for grundlæggende data og kontrolstrukturer, som kan være relevante at inddrage:

- datatyper, herunder primitive typer (såsom heltal, kommatal og boolske typer) og sammensatte typer (såsom lister eller arrays)
- variabler (både lokale og globale) samt typiske operationer på disse
- sekvenser, herunder at programinstruktioner afvikles sekventielt, og at de kan involvere variabler, der initialiseres og opdateres samt læses/skrives
- betinget udførsel (herunder boolske udtryk, sammenligningsoperatorer og sammensatte udtryk)
- løkker
- aritmetiske beregninger
- metoder til at operere på tekstvariable (såsom konkatenering samt operationer på delelementer af strenge)
- grundlæggende input og output af bl.a. tekst og tal, skrivning og læsning til filer samt
- metoder og/eller procedurer bør også indgå, så eleven har en forståelse af, at man ved brug af disse kan reducere kompleksiteten og vedligeholdelsen af koden, gøre programmet nemmere at udvikle samt øge den generelle læsbarhed og facilitere kollaborativ udvikling. Eleven bør have en forståelse af at procedurer/metoder kan have parametre og returværdier samt at parametre kan generalisere en løsning ved at benytte en procedure fremfor duplikeret kode
- klasser kan indgå på et elementært niveau, hvis det valgte sprog indbyder til det.

– *programdele og biblioteksmoduler*

Eleven bør opnå indsigt i, at konstruktionen af korrekte programmer afhænger af korrekte programdele herunder bl.a. kodesegmenter og procedurer og samtidig øves i at kombinere korrekte programdele så et korrekt program konstrueres. Det er vigtigt for eleven at erkende, at inddelingen af et program i mindre uafhængige dele kan medvirke til i højere grad at facilitere kollaborative udviklingsprocesser, øge læsbarheden af koden samt gøre det nemmere at finde fejl.
Eleven bør trænes i at inddrage og anvende eksterne biblioteksmoduler/API'er og herved blive bevidst om, at API'er bruges til at forbinde forskellige programmer, så de er i stand til at kommunikere med hinanden. Brugen af API'er og muligvis eksterne biblioteksmoduler i udviklingsprocessen kan gøre eleven bevidst om, at sådanne biblioteker kan simplificere komplekse programmeringsopgaver, øge produktiviteten og læsbarheden samt reducere mulige fejl

– *arbejdsgange og systematik i programmeringsprocessen*

Arbejdsgangene er en vigtig del af programmeringsprocessen. Eleverne vejledes til at udvikle deres programmer i en iterativ udviklingsmetode, som beskrevet i *3.1: Didaktiske overvejelser.*

Det er væsentligt, at eleven bliver bevidst om, at en programmør designer, implementerer, tester, debugger og vedligeholder programmer, når de løser et problem.

Nogle gode vaner vil også kunne støtte eleven i at beskrive programudviklingen i en synopsis, så de bliver i stand til at reflektere over samt dokumentere deres arbejde på en præcis og forståelig måde. Det er eksempelvis en hjælp at vænne eleverne til at skrive velordnede programmer med fornuftige sigende variabelnavne, anvende indrykninger, der angiver strukturen i programmet, undgå duplikeret kode og for lange kodesegmenter, samt at kommentere programmet, så andre dels kan læse det, men også vil være i stand til at arbejde videre på koden. Det er med andre ord væsentligt at betone overfor eleven vigtigheden af systematisk dokumentation. Det gælder bl.a. i forhold til at facilitere kollaborative udviklingsmiljøer, samt at ens programmeringsstil kan påvirke hvor let, det er at afgøre korrektheden af programmet.

Eleverne bør øves i at debugge deres program - dvs. lokalisere og rette fejl og herved erkende, at viden om, hvad programmet formodes at gøre, er påkrævet for at kunne finde flest mulige fejl i programmet.

– *abstrakte programmeringsbeskrivelser og dokumentation*

For den enkelte elev er det centralt at erkende, at programmerbare enheder kun gør det, de er programmeret til. Gode eksempler fra hverdagen kan være med til at anskueliggøre dette. Eksempelvis en timer til kaffemaskinen eller et trafiklys.

Et vigtigt element er at kunne læse og forklare enkle programmers virkemåde, så de kan analysere egne programmer og undgå fejl. Eleven skal altså kunne skelne mellem et programs statiske opbygning og dynamiske opførsel, som f.eks. giver sig udtryk i den korrekte implementering af en løkke, og dens virkemåde under programafvikling. Dialogen mellem lærer og elev og eleverne imellem er her et vigtigt element, der kan støtte eleverne i senere abstrakte beskrivelser af programmer, ligesom bl.a. pseudokode og flowchart støtter udviklingen af programstrukturen.

Dokumentation af programkomponenter såsom programbiblioteker, kodesegmenter, procedurer og eksterne biblioteker kan hjælpe eleven til at udvikle korrekte programmer, der løser et problem. Ved eksempelvis at benytte modeller, pseudokode, flowdiagrammer og/eller problemtræer i udviklingsprocessen kan eleven blive bevidst om, at et programs funktionalitet bedst beskrives i et højniveau sprog (f.eks. ved at beskrive hvorledes brugeren interagerer med programmet) og ikke i et lavere niveau, der er karakteriseret ved at forklare, hvordan de enkelte instruktioner i programmet fungerer.

Synopsen, som eleven udarbejder til eksamen, er med til at tydeliggøre for eleven, at det er vigtigt under programmeringsfasen at føre notater/logbog over de processer, de gennemarbejder.

2.3. Supplerende stof

Eleverne vil ikke kunne opfylde de faglige mål alene ved hjælp af kernestoffet. I forhold til de faglige samspil med de øvrige fag i uddannelsen vælges der supplerende stof med henblik på at bibringe faglig fordybelse og styrke toningen af kernestoffet. Dele af det supplerende stof vælges i samarbejde med eleverne, når det er muligt

Det supplerende stof skal opfattes som fordybelsesområder, der ligger i forlængelse af den øvrige undervisning. Eksempelvis kan nævnes programmering af en mobilapp, et spil, en webapplikation, en robot, en databaseapplikation eller et Internet of Things produkt. Programmering har mange muligheder for at indgå i tværfaglige sammenhænge med forskellige fag, hvor der naturligt kan indgå programmeringsprocesser.

Udviklingen inden for it går hurtigt og mange nye områder ser dagens lys hvert år. Mobiltelefoner, netværksteknologier, 3D mv. er nogle af disse. I den udstrækning, der er praktisk mulighed for det, kan det supplerende stof også kombineres med studiebesøg hos it-virksomheder, laboratorier og videregående uddannelsesinstitutioner.

2.4. Omfang

Det forventede omfang af fagligt stof er normalt svarende til 90 – 150 sider

Fagligt stof i faget omfatter alt fra netbaserede tutorials (herunder video-tutorials), netbaserede udviklingsværktøjer, biblioteksmoduler, dokumentationer og vejledninger, i- og e-bøger og traditionelle undervisningsmaterialer i form af bøger, udleveret tekst materiale mm.

Omfanget af fagligt stof anføres i beskrivelsen af den gennemførte undervisning (undervisningsbeskrivelsen), der færdigredigeres ved afslutningen af undervisningen i det enkelte fag. Omfanget angives normalt med en sådan detaljeringsgrad, så det af undervisningsbeskrivelsen fremgår, hvorledes det faglige stof har været vægtet i undervisningsforløbet. Dette kan fx ske ved at angive et skønsmæssigt sidetal eller en procentvis fordeling af stoffet. Opgivelsen af omfang har til formål at sikre den faglige kvalitet, så eleverne hverken under- eller overbelastes fagligt. Der kan være stor forskel på sværhedsgraden af materialerne. Derfor er der tale om en kvalificeret vurdering på baggrund af omfang og sværhedsgrad, når sidetal optælles. Er der store niveauforskellige i klassen, er det muligt at give ekstra materialer til de elever, der udviser særlig talent eller overskud.

Det kan være en god øvelse at overveje fordybelsestiden til forberedelse af et 2 siders dokument med tekst, koder, modeller osv. sammenholdt med en forberedelse af eksempelvis en videotutorial over samme tema og faglige indhold.

3. Tilrettelæggelse

Som studieretningsfag indgår faget på linje med de øvrige fag i planlægningen af den samlede studieretning, og, som nærmere uddybet i afsnit 3.4, er der adskillige muligheder for at faget kan virke katalyserende for undervisningen i flere fag.

Som valgfag er de tværfaglige samarbejdsmuligheder af naturlige grunde færre, men en differentieret undervisningsplanlægning kan gøre det muligt at finde tværfaglige samarbejdspartnere. Endelig er formaliseret tværfagligt samarbejde ikke altid en nødvendig forudsætning for, at den enkelte elev kan erkende et samspil mellem fagene. Gennem undervisningsdifferentiering kan eleven, eller en gruppe af elever, arbejde med tværfaglige

opgaver og projekter. Som oplagte eksempler på tværfaglige samarbejder nævnes teknikfag ift. programmering af mikrocontrollere eller i matematik ift. numeriske beregninger og matematisk modellering.

3.1. Didaktiske principper

– *Undervisningen tilrettelægges ved brug af anerkendte didaktiske principper, herunder 'use-modify-create'-progression fra at anvende udleverede programmer til at modificere disse for til sidst selvstændigt at skabe (nye dele af) programmer; 'Stepwise Improvement', som teknik til trinvis, iterativ og systematisk udvikling af programmer og 'Worked Examples' (kombineret med 'faded guidance'), til illustration af eksemplariske løsningsprocesser.*

Læreplanen udelukker ikke anvendelse af andre didaktiske tilgange til programmeringsundervisningen.

Stepwise improvement (fig. 1) er et eksempel på en didaktisk- og metodisk tilgang til arbejdet med programmering. For alle projektforløb gælder at selve processen med fordel kan brydes ned i flere enkeltelementer, i starten med en høj grad af lærerstyrede elevarbejder med gennemprøvede eksempler (vejledninger, tutorials m.m.) og Worked Examples (WE), og senere skal eleverne gradvist overtage processen med større grad af selvstændighed.

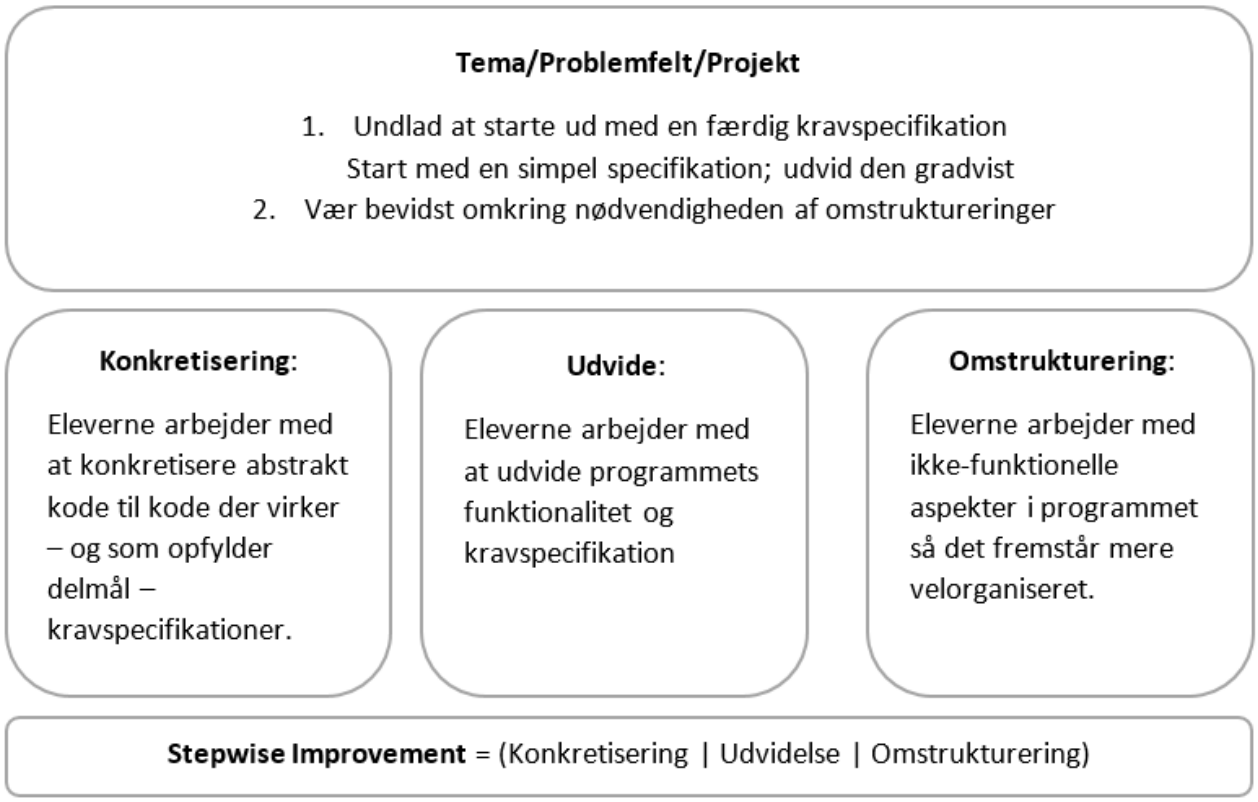

Fig 1

Eleverne skal gradvist kunne overtage processen med egen produktion (fig. 2), dels gennem forbedring og løsning af konkrete delopgaver i deres projekt med basis i de gennemprøvede eksempler (WE), dels gennem arbejdet med at udvide kravspecifikationerne til produktet (udvide) og til den færdige produktion (omstrukturere).

Modellen kan bruges som et planlægningsværktøj til hvordan man kommer fra A til B til C, og som sådan er modellen ret lavpraktisk. I stedet for at gå mere eller mindre tilfældigt frem mod et færdigt program, kan eleverne bevæge sig mere systematisk i 3 dimensioner ved dels at forbedre deres eksisterende programmer (f.eks. rette fejl), eller udvide dem (tilføje mere funktionalitet) eller omstrukturere (dvs. ændre på strukturen i deres programmer).

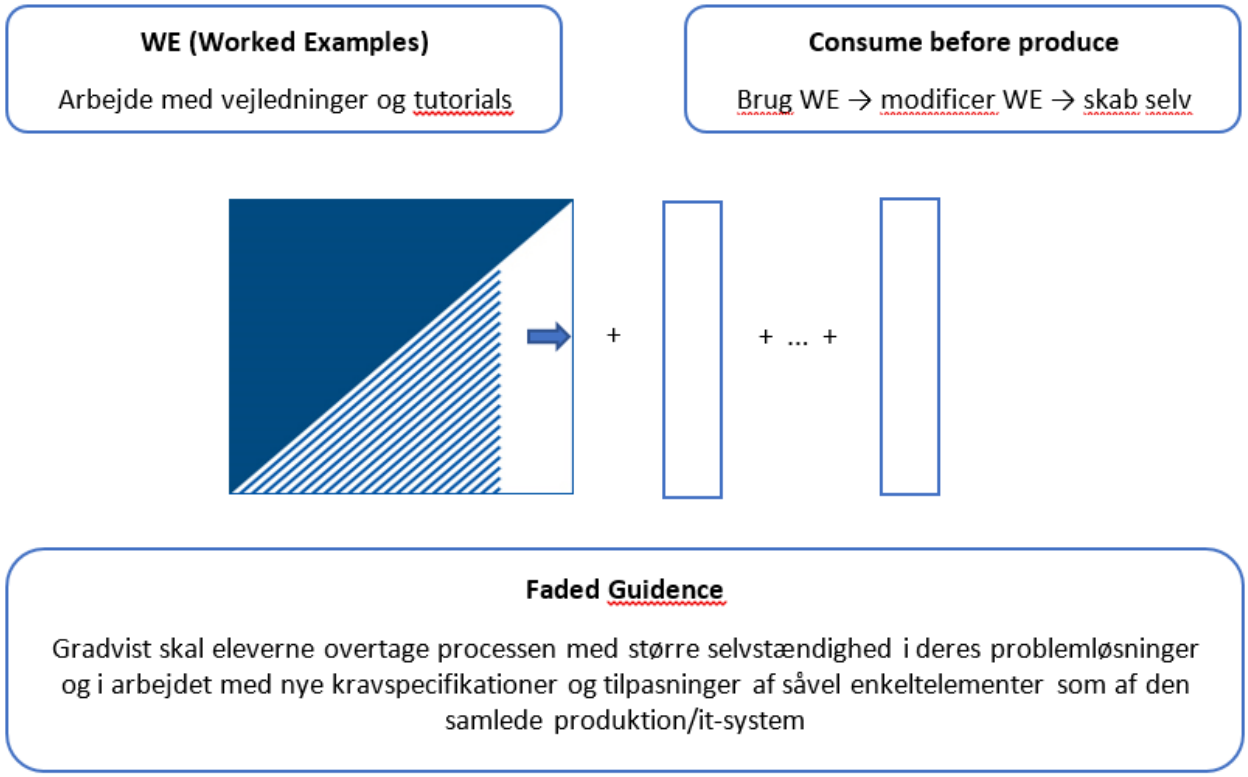

Fig 2

– *Undervisningsformen differentieres således, at alle elever udvikler sig i undervisningsforløbet. Der veksles mellem overbliksskabende forløb, eksperimenter, øvelser og projekter.*

Erfaringsmæssigt har eleverne vidt forskellige programmeringsmæssige forudsætninger ved starten af forløbet, og undervisnings-differentiering er et vigtigt redskab til at fastholde en tilstrækkelig individuel progression. Differentieringen kan eksempelvis ske gennem udstrakt inddragelse af eleverne i undervisningen gennem valg af problemstillinger, opgaver, eksempler, elevoplæg mv. Elever med erfaring kan udnyttes som en vigtig ressource for undervisningen i programmering

3.2. Arbejdsformer

– *Undervisningen tager udgangspunkt i elevens hverdag og teknologier, der indeholder programmerede funktioner. Der lægges vægt på, at eleven kan beskrive programmers funktion i normalt sprog og opnå en naturlig tilgangsvinkel til at omsætte disse funktioner til elementer i et programmeringssprog.*

Programmeringsfaget på HTX bør starte ud med at knytte an til elevernes daglige omgang med teknologi. Undervisningen bør antage en sådan karakter at eleverne får en forståelse

for fagets relevans og anvendelighed. Det er meget praksisnært, og i starten vil der være et overvejende fokus på at analysere kodeeksempler.

– *Der vælges et primært programmeringssprog som grundlag for undervisningen. Der arbejdes med eksempler på programmer. Udviklingen af forståelsen af programmeringssproget sker ved at eksperimentere med varianter af enkle programmer. Arbejdet med aktuelle teknologier skal sætte eleverne i stand til at reflektere over egne evner og interesse for karriere inden for programmering eller andre fagområder, hvor programmeringskompetencen er relevant.*

Det er vigtigt at have for øje, at den initierende læringskurve for programmering kan være meget stejl da det ofte kan være ret abstrakt for eleverne. Derfor er en blød overgang for analyse og beskrivelse af kodeeksempler over i bearbejdning/tilretning af simple programmer at foretrække. Man kan med fordel starte ud med at gøre eleverne trygge ved værktøjerne og den grafiske brugerflade i deres udviklingsmiljø. Efterfølgende kan man fx bevæge sig over i sprogets syntaks og de vigtigste kontrolstrukturer. Her kan der med fordel henvises til didaktikafsnittet og tankerne vedr. ”Use - Modify - Create”.

Skriftligheden bør komme i spil allerede her i starten. F.eks. kan der være en fordel i at lære eleverne at beskrive kode med flowdiagrammer el. ret hurtigt i deres forløb. På den måde kan de løbende træne at formidle eksekveringen af kode og deres forståelse af kode. Skriftligheden og kommunikation af programkode er en central del af programmeringsfaget og kommer i sidste ende til udtryk i deres synopsis og logbog.

– *Undervisningen tilrettelægges om muligt med udadrettede aktiviteter og/eller i samarbejde med eksterne parter, som eksemplificerer fagets anvendelses - og karrieremuligheder.*

Foruden at være et digitalt dannende fag så er programmeringsfaget også meget erhvervsorienteret. Faget sikrer, at eleven både får tekniske kompetencer og kompetence til at formidle software til forskellige målgrupper. Begge kompetencer er nødvendige i en karriere inden for softwareudvikling, hvad enten man selv er udvikler eller skal arbejde sammen med udviklere generelt. For at give eleven en realistisk forventning til, hvordan programmering indgår i erhvervslivet (både teknisk og formidlingsmæssigt), er det vigtigt, at de projekter, der laves, retter sig ud af mod samfundet. Det er ikke et krav, at der skal være samarbejde med en erhvervspartner, men det vil give en meget klar profil til faget samtidig med, at det vil skabe nogle stærke samspilsmuligheder med fx. teknik, teknologi eller kommunikation og it.

– *Undervisningsformen fremmer en progression i både indholdsmæssig sværhedsgrad og selvstændighed i problemløsningen. Der udarbejdes produkter med tilhørende dokumentation i form af synopser, herunder et eksamensprojekt.*

I løbet af undervisningen bliver eleven introduceret for nye og mere komplekse teknikker, hvor der stadig er fokus på skiftet mellem at analysere, tilrette og skabe programmer. Jo længere, man kommer i forløbet, desto større vægt vil der komme på elevernes evne til at skabe kode selv, og dermed styrkes deres innovationskompetence. Hver gang eleverne udarbejder programmer skal der også udarbejdes skriftlig dokumentation. Både i henhold til at optræne elevernes kompetencer inden for kommunikation af programmering, men også for at styrke elevernes studieforberedende skrivekompetencer. Skriftligheden trænes primært gennem synopserne, der har deres egen progression ift. indholdet og kompleksiteten. Skriftlighedens progression kan også ses endnu mere lavpraktisk hvis man starter med fx

storyboards for enkelte dele (måske eksisterende kode) og efterfølgende bevæger sig over i flowdiagrammer.

Elevernes dokumentation og strukturering af kode skal være udtryk for deres arbejde med at kommunikere software til forskellige målgrupper, dvs. på varierende grader af kompleksitet. Det er endvidere noget der giver dem både studie- og karrierekompetencer. Kompetencerne trænes både når der skrives synopsis og logbog.

– *I den afsluttende periode af undervisningen afsættes 20 timers undervisningstid til, at eleverne med vejledning fra læreren udarbejder et eksamensprojekt i grupper på to til tre. Hvor dette ikke er muligt eller ønskeligt, kan man lade eleverne arbejde individuelt. Eksamensprojektet består af et produkt og en synopsis. Synopsen skal dokumentere udviklingen af det færdige produkt og har et omfang på 5-8 normalsider, eksklusiv koder, rutediagrammer, bilag mm.*

For nogle elever er det meget naturligt at indgå i gruppearbejde, når der er tale om udvikling af programmer, mens det for andre kan være en meget vanskelig opgave. Beslutningen om at gå i grupper bør derfor nøje justeres efter dialog med underviseren, så man sikrer, at alle eleverne i en gruppe kan (og vil) bidrage til softwaren. Dette er bl.a. på grund af den stigende grad af kompleksitet i projekterne.

Når eleverne vælger arbejdsgrupper, bør de være opmærksomme på at få delt opgaverne ligeligt, så hvert gruppemedlem er klar over, hvordan vedkommende skal bidrage til softwareløsningen.

Det er vigtigt at eleverne lærer arbejdsmetoder til at arbejde sammen i grupper, fx. at bruge repositories, fx. direkte via Unity3D Cloud eller via et GIT system. På C-niveau er behovet for introduktionen til GIT ikke så nødvendigt som på B-niveau (igen grundet kompleksiteten), men det kan i mange tilfælde gøre elevernes arbejde meget lettere. Bl.a. kan introduktion til GIT være meget givtigt, hvis eleverne ønsker at gøre brug af deres programmeringskompetencer i andre fag hvor der arbejdes projektorienteret.

Produktet, der omtales, er selve softwareløsningen, og synopsen, der omtales, er en skriftlig opgave, hvor eksaminandens problemstilling opridses, og hvor hele processen fra planlægning over design til udvikling og afprøvning dokumenteres. I synopsen inddrages diagrammer og andre visualiseringer, der kan visualisere elevernes tanker om strukturerингen af deres software. Endvidere gennemgås udvalgte dele af softwaren på kodeniveau.

Synopsens indhold og form beskrives i afsnittet “Eksamensprojektets synopsis er [...]”.

– *Eksamensprojektet udarbejdes inden for rammerne af et projektoplæg stillet af skolen. Eksamensprojektgrupperne udarbejder en fælles projektbeskrivelse, der inkluderer en beskrivelse af den enkelte eksaminands fokus. Projektbeskrivelsen godkendes af skolen, når beskrivelsen er tilstrækkelig fagligt bred og niveaumæssigt relevant.*

I eksamensprojektet vælger eleverne deres problemstilling ud fra det projektoplæg som underviseren udarbejder. Projektoplægget skal være klart formuleret og give en grundig beskrivelse af en problemstilling. Det bør endvidere have en sådan karakter, at eleverne kan relatere sig til det både indholdsmæssigt og fagteknisk. Et grundigt formuleret oplæg bør også være så fleksibelt, at forskellige elev-kompetencer kan komme i spil. Det er vigtigt, at man igennem hele fagets forløb sikrer en gradvis større selvstændighed i elevernes arbejde, således at de bliver i stand til at arbejde selvstændigt med eksamensprojektet.

Derfor bør eleverne inden eksamensprojektet have afprøvet projektarbejdsformen, eksempelvis gennem flere forudgående projekter af mindre omfang med god mulighed for feedback og erfaringsudveksling mellem lærer og elev og eleverne imellem.

Husk endvidere, at det i gruppernes projektbeskrivelser fremgår, hvilket ansvar den enkelte eksaminand har i projektet. Dette bør kontrolleres i forbindelse med godkendelsesprocessen.

– *Eksamensprojektets synopsis er individuelt udarbejdet. Afleveringstidspunktet skal normalt være senest en uge før eksamensperiodens begyndelse*

I synopsen inddrages der diagrammer og andre visualiseringer der kan billedliggøre elevernes tanker om struktureringen af deres software. Endvidere gennemgås udvalgte dele af softwaren på kode niveau. En synopsis skal forstås som en tekst, der skrives til eksaminator og censor som forberedelse til den mundtlige eksamen. Det vil sige, at en synopsis bør give censor og eksaminator et overblik over projektet, i form af relevante abstrakte dokumentationsformer, og passende forklaringer i almindelig tekst.

Det anbefales at stille et eksempel på opbygningen af en synopsis til rådighed for eleverne. En sådan eksemplarisk synopsis bør ikke omfatte for komplekse programmer, men gerne dække de dele, man normalt ønsker at få dokumenteret i en synopsis, så som:

- Forblad
- Kort abstract (censor kan herved orientere sig om opgavens indhold)
- Problemformulering
- Funktionsbeskrivelse (skærmlayout, indtastningsmuligheder, funktionalitet – alt efter hvad det er for et program)
- Dokumentation af selve programmet (overordnet beskrivelse af programmet, detaljeret dokumentation af dele af programmet (flowchart, pseudokode), variabler, objekter, events, igen meget afhængigt af hvad det er for et program)
- Test af programmet
- Konklusion
- Bilag (det er godt at få koden placeret i bilag, da den muligvis ikke ville kunne indeholdes indenfor journalens 10 sider).

– *Eksamensprojektet indgår i grundlaget for den afsluttende standpunktskarakter, hvis der gives en sådan, og udgør grundlaget for prøven.*
Eksamensprojektets synopsis er forinden prøven ikke rettet og kommenteret af eksaminator.

Det er vigtigt, at man sammen med administrationen på skolen sikrer, at der tid nok ved afslutningen af eksamensprojektet til at inddrage projekterne i den afsluttende standpunktskarakter. Samtidig må man ikke give eleverne feedback på synopsis og projekt inden prøven.

– *Den enkelte elev dokumenterer løbende sin faglige udvikling i en logbog. Dokumentationen i logbogen kan have form af f.eks. programmer, noter, synopser, programbeskrivelser og flowcharts.*

Inden eksamen udfærdiges en logbog. Den laves undervejs, og der laves eventuelt en portfolio i slutningen. Logbogen er en skrivegenre, hvor eleven reflekterer over egen læring. Den bør således have tilknyttet et refleksionsresumé for hvert forløb, hvor eleven gør sig nogle overvejelser om den læring der er opnået. Logbogen bør endvidere være tilgængelig for såvel elev som lærer. Det kan være på et fælles drev (fx. OneDrive, Google Drive, et skoledrev el.)

Man kan også forestille sig, at eleverne dokumentere deres arbejde via GIT, fx. GITHub.

For eleverne findes der overordnet set to typer af skriftligt arbejde. Den løbende logbogsføring og refleksionen, der grundlæggende er rettet mod eleven selv, og så er der det formidlingsmæssige arbejde (så som synopsen, der er beskrevet i afsnit: “Eksamensprojektets synopsis er [...]”.

Den skriftlige formidlingskompetence er meget tæt knyttet til elevernes studie- og karrierekompetencer, idet de videre i deres karrierer vil kunne drage nytte af at kunne formidle software på flere forskellige niveauer (teknisk) til forskellige målgrupper.

3.3. It

- *Gennem arbejdet med udvikling af programmer i faget opnås såvel specifikke faglige digitale kompetencer som almene digitale kompetencer, hvilket er fagets bidrag til uddannelsernes overordnede krav om digital dannelse.*

I faget vil eleverne stifte bekendtskab med den tekniske side af de systemer som de bruger i deres dagligdag. Indsigten i, hvordan deres programmer fra hverdagen er skruet sammen, er med til at give dem en grundlæggende forståelse for opbygningen af programmer. Denne indsigt kan de bruge til at generalisere erfaringer, når de anvender andre systemer, og samtidig når de selv udvikler nye programmer.

– *Programmeringsværktøjer, der automatisk kan generere dokumentation og debugging, anvendes, ligesom andre informationsteknologiske værktøjer inddrages efter behov.*

Her er der tale om udviklingsmiljøer (såsom Microsofts Visual Studio) eller versioneringssystemer (såsom Git)

- *Internettet anvendes som søgningsværktøj til oplysninger, vejledninger, eksempler, programdele og biblioteksmoduler med efterlevelse af ophavsretslige regler og dokumentationskrav.*

Særligt inden for programmeringsfeltet er det muligt at finde vejledninger og dele af programkode på internettet. Disse dele inddrager eleverne ofte i deres løsninger, og her er det væsentligt at tale med eleverne om, hvordan de gør programmerne til deres egne ved at ændre på dem.

– *I arbejdet med stof om konkrete teknologier og standardiseringer skal eleverne anvende originale kilder (eksempelvis dokumentation af programmeringssprog, data og diagrammer).*

Eleverne bør trænes i at slå op i relevante referencer til API’er og andre eksterne biblioteker samt dokumentere brugen heraf. Eleverne bør endvidere kunne tage udgangspunkt i flowdiagrammer, når de programmerer.

- *Eleverne arbejder med digital dokumentation af deres programmer, bl.a. i form af modeller og kommentarer i programmeringskoden*

Blandt relevante modeller kan nævnes flowdiagrammer, UML-diagrammer, pseudokode eller lignende.

3.4. Samspil med andre fag

Samspilsdimensionen er meget vigtig, bl.a. fordi det indgår i fagets faglige mål og fordrer en kvalificering af elevernes innovationskompetence.

Det hedder sig, at man skal *"løse en enkel problemstilling gennem udviklingen af et program bl.a. i samspil med andre fag"*.

Løsningen af problemstilling kan gøres på mange forskellige måder, og her er listet nogle forskellige tilgange til samspilsdimensionen, uanset hvilken uddannelse faget måtte indgå i.

Konstruerende

Faget giver selv sagt nogle tekniske færdigheder der gør eleven i stand til at producere forskellige produkter. Faget kan i en samspilssituation ses som værende produktudviklende og på den måde kan det indgå i en situation hvor et andet fag analyserer produktet og evt. forsøger at markedsføre det.

Analyserende

I faget er der mange værktøjer til at analyse en given problemstilling. Eksempelvis kan man opstille en kravspecifikation for en problemstilling der er givet i et andet fag. Kravspecifikationen kan være et udgangspunkt for en udviklingsproces, hvor man efterfølgende forsøger at beskrive hvorledes scenariet kan løses på software niveau (eksempelvis beskrevet diagrammatisk).

4. Evaluering

4.1. Løbende evaluering

Den løbende evaluering af undervisningsforløb kan kombineres med at træne eleverne i udformning af den logbog, som de laver i forbindelse med eksamensprojektet. Her kan eleverne reflektere over deres faglige udbytte af forløbet, arbejdsprocessen og de anvendte arbejdsformer, perspektivere til andre fag, osv. Suppleret med spørgsmål fra læreren om f.eks. sværhedsgrad, motivation og relevans, kan eleverne få indflydelse på kommende projekter i undervisningen.

Det er vigtigt at synliggøre kravene for eleverne i de enkelte forløb, da de som nybegyndere ikke er bekendte med fagets niveau, og derfor ikke altid kan vurdere om en opgave er svær eller let, og i hvilken grad de opfylder målene. Dette er særlig relevant når man har en høj grad af differentiering på holdet, og i høj grad lader eleverne være med til at sætte mål for deres egen læring.

Det er naturligt at man vurderer elevernes evner til at beherske det valgte programmeringsmiljø, når man hjælper eleverne i den daglige undervisning, både i den første indlæringsfase af sprogets grundelementer, samt i de mere projektorienterede forløb.

Den afsluttende evaluering i faget er baseret på et eksamensprojekt, med et afsluttende produkt og tilhørende synopsis. Eleverne bør løbende have træning i at udforme relevant dokumentation af deres arbejde, både jvf. det tilhørende kernestofområde, men

også fordi de er helt uerfarne med den skriftlige diskurs i faget. Her er det vigtigt at give løbende feedback.

4.2. Prøveform

Eksaminanden skal inden eksamensperiodens begyndelse aflevere sit eksamensprojekt i form af produkt og synopsis, som eksaminator evaluerer som forberedelse til den mundtlige eksamen.

Synopsen er forinden prøven ikke rettet og kommenteret af læreren.

Eksaminanden afleverer to eksemplarer af synopsen. Den ene afleveres til eksaminator; den anden kan skolen evt. fremsende til censor.

Skolen bør stille værktøj til rådighed for eleverne, så de kan opbevare og præsentere deres eksamensprojekt i elektronisk form. Eleverne skal løbende samle deres programmeringsarbejde i en portfolio, og eksamensprojektet kan indgå som en naturlig del af denne.

Afleveringsfristen for eksamensprojektets produkt og synopsis fastsættes af skolen, dog således at tidsfristen er senest en uge før eksamensperiodens begyndelse. Produkt og synopsis skal være til rådighed ved eksaminationen.

Det vil ofte være en fordel, at eksaminanden medbringer eget udstyr til at understøtte sin fremlæggelse. Det kan være produktet på en bærbar computer og en multimediepræsentation.

Der er afsat ca. 24 minutter til eksaminationen, og der er ingen forberedelsestid. Eksaminanden starter med at præsentere sit eksamensprojekt, med tilhørende teoretiske overvejelser, for eksaminator og censor. Eksaminators supplerende spørgsmål fungerer som oplæg til en uddybende samtale om de punkter, som ikke er berørt, eller der ikke fyldestgørende er redegjort for i synopsen.
Af hensyn til bedømmelsesgrundlaget kan de supplerende spørgsmål have deres grundlag i et eller flere af de faglige mål, som er angivet i læreplanen (Læreplanen 2.1).

Det er udelukkende elevens mundtlige præstation der evalueres til eksamen.

Adgang til internettet:
For elever og kursister, der har begyndt gymnasial uddannelse efter den 1. august 2017 (”nye” elever) gælder reglerne i den nye eksamensbekendtgørelse (§6 i Bekendtgørelse om visse regler om prøver og eksamen i de gymnasiale uddannelser).

For mundtlige prøver betyder det, at internettet som fagligt hjælpemiddel ikke er tilladt i forberedelsestiden.
Det fremgår også af den nye bekendtgørelse, at forbuddet mod brug af internettet under prøverne kun gælder for prøver, hvor eksaminanden under prøven skal være til stede på institutionen (eller et andet sted, som institutionen fastsætter for prøveafholdelsen).
For prøverne i den nye læreplan for programmering C, hvor eleverne i den afsluttende periode af undervisningen udarbejder et eksamensprojekt, og hvor der ikke er forberedelsestid i forbindelse med den mundtlige prøve, har eleverne også adgang til internettet som fagligt hjælpemiddel, når de udarbejder eksamensprojektet .

Hvis adgang til internettet er nødvendig for eksaminandens præsentation af sit eksamensprojekt, er dette tilladt under eksamination.

"Regler vedrørende eksaminandernes brug af internettet for at tilgå tilladte hjælpemidler ved prøverne fremgår af § 6 i "Bekendtgørelse om visse regler om prøver og eksamen i de gymnasiale uddannelser".
I vejledningen til denne bekendtgørelse er der givet eksempler på, hvilke hjælpemidler der må, og hvilke der ikke må tilgås via internettet."

4.3. Bedømmelseskriterier

– *Bedømmelsen er en vurdering af, i hvilken grad eksaminandens præstation opfylder de faglige mål, som de er angivet i pkt. 2.1.*

Med internettets muligheder for at finde mange gode programmer og programdele, bør det fremgå klart, hvilke dele af programmet eleven selv har fremstillet, og hvornår der er tale om andres programdele og biblioteksmoduler. Oprindelsen bør fremgå af synopsen, og eleven bør kunne dokumentere, hvordan de bruges.

I tvivlstilfælde kan man bede eleven om at ændre en lille smule på funktionaliteten af programmet. Det kan ligeledes være relevant at bede eleven redegøre for udvalgte kontrolstrukturer i programmet og argumentere for valget af netop disse frem for andre af eleven kendte kontrolstrukturer.

Det kan også være afklarende at spørge om, hvordan programmet er blevet udviklet – hvad har eleven startet med at løse, og hvordan er opbygningen ellers sket. Det kan også give et godt billede af elevens evner at få oplyst hvilke problemer, der har været i udviklingen af programmet. Især for de dygtige elever viser det noget om deres kompetencer inden for programmering, hvis de kan give forslag til hvordan problemstillingen ellers kunne have været løst.

– *Ved prøve, hvor faget indgår i samspil med andre fag, lægges der vægt på at eksaminanden*
    - *kan demonstrere viden om fagets identitet og metoder*
    - *behandle problemstillinger i samspil med andre fag*

Hvis en eksaminand eksempelvis har lavet et projekt i samspil med teknikfaget hvor eksaminanden programmerer en mikrocontroller, bør der indgå forklaring af de el-tekniske begreber og fænomener der er relevante for projektet (det kunne være et begreb som pulsbreddemodulation el.)

Oversigt over karakterskalaen

12	Fremragende	Karakteren 12 gives for den fremragende præstation, der demonstrerer udtømmende opfyldelse af fagets mål, med ingen eller få uvæsentlige mangler.
7	God	Karakteren 7 gives for den gode præstation, der demonstrerer opfyldelse af fagets mål, med en del mangler.

02	Tilstrækkelig	Karakteren 02 gives for den tilstrækkelige præstation, der demonstrerer den minimalt acceptable grad af opfyldelse af fagets mål.

4.4 Vejledende karakterbeskrivelse

Nedenstående er en vejledende karakterbeskrivelse for Programmering C valgfag for karaktererne 12, 7 og 02.

Beskrivelsen er udarbejdet med udgangspunkt i læreplanens faglige mål og bedømmelseskriterier

Karakter	**Beskrivelse**	**Programmering C valgfag**
12	Fremragende	Eksamensprojektet præsenteres glimrende og fagligt sikkert mht. planlægning, gennemførsel og evaluering. Eksamensprojektet lever op til de stillede krav i opgaven med kun få uvæsentlige mangler. Der argumenteres fagligt velbegrundet for valg af faglige teorier og metoder. Eksaminanden perspektiverer fagligt kvalificeret sin viden til såvel egne programmerings-løsninger som til opgavens teoretiske indhold. Eksaminanden besvarer glimrende og fagligt sikkert uddybende og supplerende spørgsmål under samtalen.
7	God	Eksamensprojektet præsenteres mht. planlægning, gennemførsel og evaluering. Eksamensprojektet lever trods en del mangler op til de stillede krav. Der argumenteres for valg af faglige teorier og metoder. Eksaminanden perspektiverer sin viden til såvel egne programmerings-løsninger som til opgavens teoretiske indhold. Eksaminanden besvarer uddybende og supplerende spørgsmål under samtalen.
02	Tilstrækkelig	Eksamensprojektet præsenteres sparsomt og knapt mht. planlægning, gennemførsel og evaluering. Eksamensprojektet lever minimalt acceptabelt op til de stillede krav. Der argumenteres minimalt acceptabelt for valg af faglige teorier og metoder. Eksaminanden perspektiverer tilstrækkeligt sin viden til såvel egne programmerings-løsninger som til opgavens teoretiske indhold. Eksaminanden besvarer sparsomt og knapt uddybende spørgsmål under samtalen.