# First Program

|  |  |
| --- | --- |
| **Læringsmål** | Du kan:   * 1Pf1: anvende centrale metoder til at specificere og konstruere algoritmer [...] * 1Pf2: anvende centrale faciliteter i programmeringssproget til realisering af algoritmer […] * 1Pk3: i en struktureret sammenhæng tilegne dig ny viden, færdigheder og kompetencer inden for […] udviklingsværktøjer […] |
| **Forventede  produkter** | * En beskrivelse af en studerendes dagligdag via Computational Thinking tilgangen (da: systemisk tænkning). * Et C#-konsolprogram, der kan udskrive tekst til samt indlæse tekst fra konsolvinduet. |
| **Forventet læringsudbytte [SOLO]** | Designsporet:   * [Unistrukturel] Du kan nævne enkelte kendetegn ved **Computational Thinking**   Programmeringssporet:   * [Unistrukturel] Du kan genkende enkelte kendetegn i den **logiske struktur** af en applikation (kodeblok, sætning). * [Unistrukturel] Du kan genkende enkelte kendetegn ved **datarepræsentation** (variabel-erklæring) * [Unistrukturel] Du kan genkende enkelte kendetegn ved **datatyper** (string) * [Unistrukturel] Du kan genkende enkelte kendetegn ved **datahåndtering** af en variabel (tildeling af værdi til en variabel, læsning af en variabels værdi) * [Unistrukturel] Du kan genkende enkelte kendetegn ved **program flow** (sætning, sekvens) * [Unistrukturel] Du kan genkende enkelte kendetegn ved en **konsolapplikation** (Console.WriteLine, Console.Write og Console.ReadLine) |
| **Din forberedelse** | **Bemærk**: Det er ikke al materiale til denne opgave, som nødvendigvis skal læses/ses, da det er op til dig selv at vælge, hvor meget (eller hvor lidt), du vil se, afhængig af den tid, du har til rådighed. Dette er et forsøg på at differentiere lidt i forberedelsen.  Som minimum til opgavens løsning, da læs tekst eller gennemse videoer markeret med **fed tekst** (eller start med dem). Det forventes, at du hjemme har gennemgået minimumsmaterialet, inden du møder op til undervisning, så du er parat til at løse opgaven sammen med dit team.  Nogle videoer er i forberedelsen angivet med et link (f.eks. til youtube).  Videoer uden et link kan i stedet findes i Planer i ItsLearning.  Se videoerne og læs teksterne i den rækkefølge, der er angivet forneden.  Læring:   * **Cooperative Learning (CL): Møde-på-midten og (B)Ordet-rundt  (se i ItsLearning i folderen ’1. Studieår’ --> ’Cooperative Learning Strukturer’)**   Designsporet:   * [**What is Computational Thinking**](https://www.youtube.com/watch?v=GJKzkVZcozc) **(video: 5:37)** * [**Computational Thinking: What Is It? How Is It Used?**](https://www.youtube.com/watch?v=qbnTZCj0ugI) **(video: 5:41)** * [**Introduction to Computational Thinking (BBC)**](https://www.bbc.co.uk/bitesize/guides/zp92mp3/revision/1)   Programmeringssporet:   * **[YB] 1.1-1.2 (s. 2-9, 8 sider)** * **Opret nyt C#-projekt (video: 7:04)**   + Den fysiske struktur af et C#-projekt (video: 5:53)   + Den logiske struktur af et C#-projekt (video: 6:06) * Skriv/Byg/Kør et C#-projekt (video: 9:35) * Programflow – sætning og sekvens af sætninger (video: 3:50) * **String – literal værdi (video: 5:29)** * **Skriv tekst til konsolvinduet (video: 9.04)** * **String – konkatenering (video: 7:26)** * **Læs tekst-input fra konsolvinduet (video: 8:15)** * **Debugging – Step Over (video: 7:10)** * **Intro til variable (video: 9:11)** |

# Dagens ord: Den gode start

Måske tænker du, at du blot skal have adgang til den rette ressource, den perfekte bog, en dygtig underviser, medstuderende, lektiecafé-hjælper, eller måske skal du blot se en række af videoer omkring det at programmere – og så skal du nok blive god til det.

Alt dette kan hjælpe, men det må ikke stå alene.

At have en god studie-/arbejdsrutine er altafgørende for, om du med tiden bliver dygtig til at programmere, eller om du ender med at vide en masse, men intet kunne.

Det kan ikke påtales nok, men det med at forstå logikken, der er beskrevet i en bog, eller at kunne følge med i et praktisk eksempel i en video, det er typisk den ’lette del’.

Hvis du vil have **færdigheden**, så skal du have det ind under neglene; du skal lave hjemmeprojekter, og du skal undgå at opsøge et direkte svar, men du skal i stedet søge hjælp til at komme videre (det er bestemt ikke forbudt at opsøge hjælp, men prøv først selv i 5-10 minutter, så vil du være i stand til at huske løsningen en anden gang).

*Så hvad er et hjemmeprojekt?* Et hjemmeprojekt kan være blot en lille stump kode, noget nyt, som du har lært i løbet af dagen i dag, sat i en ny kontekst – eller det kan være flere ’dele’, der til sammen løser en større problemstilling. Det tager ikke mange minutter ud af hver dag at etablere en god rutine, så start med at få en god studie-/arbejdsrutine allerede nu.

# Øvelse 1: Terminologi

I skal reflektere over begreberne ”Computational Thinking”, ”applikation”, ”instruktion”, ”sætning (en: statement)” og ”kodeblok”, med udgangspunkt i følgende beskrivelse:

1. Hvert enkelt medlem i gruppen skriver sine overvejelser om begreberne ned (2 minutter).
2. Hver gruppe starter **Ordet rundt**, idet det første gruppemedlem forklarer sine overvejelser   
   (2 minutter). Hele gruppen diskuterer bidraget (2 minutter). Hvis gruppen enes om, at det er en god forklaring, skriver det første gruppemedlem sit bidrag ind i gruppens dokument.
3. Det næste gruppemedlem forklarer sine overvejelser osv. Fortsæt med **Ordet rundt** indtil alle gruppemedlemmer har bidraget, eller tiden er gået.

*Tidsramme: 30 minutter*

# Øvelse 2: Computational Thinking

Står man over for et problem, der skal løses med software, så nytter det ikke at sætte sig foran computeren og begynde at kode. Før man går i gang med at kode, skal problemet først forstås, nedbrydes, omformes, formaliseres, så det kan udtrykkes i en algoritme og derefter kodes i et computerprogram som en løsning, der giver værdi for brugeren. Rob Miles i Yellow Book [YB] udtrykker det således i to trin:

1. You need to be able to solve the problem yourself   
   before you can write a program to do it.  
   2. The computer has to be made to understand what you are trying to tell it to do.  
   [Yellow Book, s. 5 øverst, Rob Miles]

De næste øvelser giver dig mulighed for at arbejde med de fire nøglebegreber bag **Computational Thinking**, så du er lidt mere bevidst om processen at afklare et problemfelt, finde de centrale elementer, nedbryde problemet og specificerer en algoritme, inden du starter med at kode en softwareløsning.

## Øvelse 2.1: Daglig rutine for UCL-studerende

Du skal sammen med dit team opbygge en algoritme, som beskriver den daglige rutine for en typisk datamatiker-studerende på UCL.

Brug **Ordet Rundt** i hver gruppe, og find frem til de forskellige **studierelaterede aktiviteter** en studerende har på forskellige tidspunkter af dagen, både når man er på UCL og privat. Overvej, hvad der er på den **daglige tidslinje** (vigtige tidspunkter og aktiviteter):

* Abstrahér fra det personlige og find fællestrækkene, så algoritmen passer på alle.
* Bemærk, at nogle gange gør man kun noget bestemt, hvis en betingelse er opfyldt (Hvis ”det her” gælder, da gør ”det her”); tag dette med i jeres overvejelser. Husk at nogle aktiviteter sker på bestemte tidspunkter af dagen.
* Opbyg og nedskriv en algoritme i punktform ud fra det, I har fundet frem til i din gruppe. Tænk produktet som en instruktion, man kan give til en UCL-studerende, der kan følge alle dagens trin.

*Tidsramme: 20 minutter*

## Øvelse 2.2: Præsentation af jeres algoritme

Nu skal I fremvise jeres algoritme, samt se, hvad de øvrige teams er nået frem til.

Fremgangsmåde:

* Klæd en studerende i gruppen på til at kunne fremlægge jeres algoritme til de øvrige teams (5 minutter)
* En bliver tilbage og fremlægger jeres algoritme, de øvrige går på tur i BrainBoxen, og ser, hvad de øvrige studerende er nået frem til (20 minutter)
  + I tager noter imens I er på tur, samt stiller spørgsmål *(I skal bruge disse noter I jeres eget team, samt I skal se, hvordan de øvrige teams har forstået selv samme øvelse*)

*Tidsramme: 25 minutter*

# Øvelse 3: Dit første program

Formålet med denne øvelse er at få en indledende føling med, hvad en applikation er inkl. trinene med at oprette et projekt (konsolapplikation) og køre applikationen; dvs. at få det konkret i hænderne på din computer. Applikationen er meget simpel og har fokus på ganske enkel brugerinteraktion. Samtidig kommer du til at stifte bekendtskab med forskellige termer for flere grundlæggende fagbegreber.

Arbejd sammen med din sidemakker to og to, mens I gennemfører øvelsen (denne samarbejdsform hedder **parprogrammering**, som du kommer til at høre mere om senere). Hvis I er et ulige antal, da kan I også arbejde tre sammen.

**Note:** Hvis du **ikke har fået installeret** Visual Studio IDE på din computer endnu:

* Benyt <https://dotnetfiddle.net/> i de følgende øvelser 3.2, 3.3 og 3.4 (spring 3.1 over)
* Betragt øvelse 3.5 (debug) som forberedelse til næste gang

## Øvelse 3.1 Oprettelse af projekt i Visual Studio

Om lidt skal du til at lave din første applikation, men før du kan komme i gang med at skrive kode, skal du vælge en passende skabelon inde i Visual Studio, når du opretter projektet. På nuværende tidspunkt (og den kommende tid) skal du ikke fokusere på det visuelle aspekt, du skal først have styr på det helt basale. Derfor skal dit første C#-projekt oprettes som en **konsolapplikation**.

Udfør følgende:

* Start Visual Studio
* Opret en ny konsolapplikation med skabelonen: “Console App” (C#). Vælg selv projektnavn.
* Kør den autogenererede kode i applikationen, så du sikrer dig, at alt fungerer fint.

Du skal i det følgende tilrette denne kode.

## Øvelse 3.2: Din første C#-applikation

Du skal sammen med din sidemakker udvikle en applikation, der kan skrive navn og alder ud til konsol­vinduet. Du skal øve dig i et skrive flere C#-sætninger (en: statements), der udføres sekventielt (dvs. i rækkefølge) efter hinanden.

**Bemærk:** **Skriv i denne øvelse udelukkende sætninger inde i kodeblokken for Main()** dvs. mellem de to tuborgklammer { } og ingen andre steder.

Udfør følgende (læs nedenstående omhyggeligt):

* Fjern den autogenerede linje, der skriver ”Hello, World” ud til konsolvinduet.
* Tilføj en ny linje, og skriv en sætning (en: statement) med teksten ”Navn: ” ud til konsolvinduet (husk mellemrummet efter kolon)
* I en ny sætning skriv derefter på samme linje et navn ud til konsolvinduet, f.eks. ”Jens Hansen”
* På en ny linje, skriv teksten ”Alder: ” ud til konsolvinduet
* Igen på samme linje skriv en alder ud, f.eks. ”25”

Du skulle nu gerne have 4 sætninger fordelt på fire linjer i koden, men når du kører programmet, skal du gerne kun se følgende 2 linjer øverst i konsolvinduet:

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

Hvis det ser anderledes ud, så ret i de 4 sætninger, så det passer. Passer det? Tillykke! Du har nu præsenteret tekst for brugeren (dig selv) i din første C#-applikation.

## Øvelse 3.3: Brugerinput

Det næste trin er at give din applikation mulighed for at modtage tekst fra brugeren, dvs. modtage brugerinput. I stedet for at angive navn og alder i selve koden (dvs. ”Jens Hansen” og ”25”), da skal du indlæse disse to oplysninger direkte fra konsolvinduet.

Tilret programmet fra øvelse 3.2:

* Så den første sætning skriver teksten ”Indtast navn: ” ud til konsolvinduet
* Skriv dernæst ”Navn: ” efterfulgt af den indlæste tekst fra brugeren ud til konsolvinduet   
  Vink: benyt ReadLine() sammen med WriteLine()
* Skriv teksten ”Indtast alder: ” ud på en ny linje i konsolvinduet
* Skriv dernæst ”Alder: ” efterfulgt af den indlæste tekst fra brugeren ud til konsolvinduet

Du skulle igen ende ud med 4 sætninger i koden, men når du kører applikationen, burde du nu gerne kunne se følgende 4 linjer:

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

## Øvelse 3.4: Navn og alder med variabler

Du skal nu til sidst i konsolvinduet (på den 5. linje) skrive en tekst med formuleringen:

* ”Jens Hansen er 25 år gammel”

Dvs. du har brug for at sammenkæde (en: concatinate) en tekststreng med bidrag fra begge indlæsninger (navn og alder) plus noget ekstra tekst (”er” og ”år gammel”).[[1]](#footnote-1)

Fremgangsmåden i forrige øvelse med brug af ReadLine() er problematisk, da du ikke gemmer den indtastede information. Derfor skal du nu tilrette koden.

Udfør følgende (ved at tilrette koden fra forrige øvelse de rette steder):

* Erklær en variabel af typen string med navnet ’name’, der kan indeholde et navn
* Erklær en variabel af typen string med navnet ’age’, der kan indeholde en alder
* Tildel værdier til de to variabler via passende ReadLine()-kald
* Udskriv til konsolvinduet (via reference til de to variable ’name’ og ’age’) formuleringen:  
   ”Jens Hansen er 34 år gammel”, hvis det indtastede navn er ”Jens Hansen”, og tilsvarende alder er ”34”. Husk mellemrum.

Du skulle gerne ende med at se følgende i konsolvinduet:

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

## Øvelse 3.5: Debugging

I denne øvelse skal du for første gang afprøve debuggeren i Visual Studio på den kode, du lige har skrevet. Debugging er en proces, hvor du trinvis kan inspicere din kode, mens koden kører, og du kan lokalisere uhensigtsmæssigheder (nogle gange også fejl) i din kode, med henblik på at udbedre disse uhensigtsmæssigheder.

Udfør følgende:

* Benyt debug-kommando ’Step Over’ (eller funktionstasten F10) til trinvis at udføre din kode fra øvelse 3.4. Inspicér konsolvinduet efter hvert trin, så du kan se, hvad der er skrevet ud på et vilkårligt tidspunkt.

Hint: Det kan være en god idé, at se om der er noget der kan benyttes fra forberedelsen.

# Øvelse 4: Review spørgsmål

Du og dit team skal benytte en passende CL-struktur til at gennemgå følgende spørgsmål:

* Hvordan angiver man en tekststrengsværdi?
* Hvordan udskriver man noget til konsolvinduet?
* Hvordan modtager man brugerinput fra konsolvinduet?
* Hvad er en variabel?
* Hvad vil det sige at debugge sin kode?

*Tidsramme: 15 minutter*

1. Når man sammenkæder to tekststrenge, kaldes dette for “concatination” på engelsk [↑](#footnote-ref-1)