**Algoritmer**

Formålet med dagens opgave er at forstå og beherske nogle algoritmer, der er afgørende for at bygge effektive og optimerede applikationer. I denne opgavesæt vil vi dykke ned i tre typer algoritmer og undersøge, hvordan man implementerer dem i C#. De tre typer algoritmer er sortering, søgning og rekursion.

Dagens opgave er udviklet på en måde til at du genbesøger nogle emner, du blev introduceret tidligere såsom klasser, objekter, datastrukturer, pseudokode og filbehandling.

# Du skal i dag gå sammen med en anden i gruppen og benytte parprogrammering (Ja, det vil sige, at én er driver og én er navigatør/observer).

# Øvelse 1: Terminologi

Inden du skal i gang med dagens konkrete programmeringsopgaver, skal du have testet din forforståelse af dagens emne.

Del teamet op i 2 grupper, og brug CL-strukturen **tænk-par-del** (se nedenstående fremgangsmåde) til at reflektere over begreberne:

1. *Algoritm, sortering algoritm, søgning algoritm, rekursion.*
2. *Kort beskrivelse af selection sort, bubble sort og quick sort algoritmer. Brug evt. et whiteboard til at visualisere hvordan virker de og hvad er forskellen mellem dem.*
3. *Kort beskrivelse af linear search og binary search algoritmer. Brug evt. et whiteboard til at visualisere hvordan virker de og hvad er forskellen mellem dem.*
4. *Kort beskrivelse af rekursion og dens fordele og ulember.*

**Fremgangsmåde:**

1. Team: Del punkterne ud mellem de to grupper (1 minut)
2. Individuelt: Skriv dine overvejelser ned om begreberne (5 minutter)
3. Par: Del jeres tanker med hinanden i gruppen (2 minutter)
4. Par: Forbered jer på, hvad der skal deles (2 minutter)
5. Team: Præsentere på skift det, som I er blevet enige om, til hele teamet (uret rundt) (2 minutter per gruppe)
   1. De øvrige omkring bordet tager noter og stiller spørgsmål

*Tidsramme: 15 minutter*

## Øvelse 2: Implementering af Student klasse

Benyt parprogrammering til at implementere Student klasse.

* Indsæt en ny C# klasse ”Student” (i en separat C#-fil) til konsolprojektet og gør klassen public.  
  C#-klassen skal udtrykke følgende designklasse:

|  |
| --- |
| Student |
| **+ <<property>> FullName: string**  **+ <<property>> GroupNumber: int** |
| + <<constructor>> Student(fullName: string, groupNumber: int) |

Klassen angiver et antal properties, som samlet udgør klassens interne datastruktur. Bemærk, at alle properties, er angivet som public i klassen (plustegnet) og starter med stort bogstav (UpperCamelCase).

Udfør følgende:

* Implementér designklassens to properties i C#-klassen for Student.
* Definer constructor i klassen med den angivne header, som initialiserer alle objektets properties med parametrene: fullName og groupNumber.

## Øvelse 3: Filbehandling

Studerendes oplysninger du kommer til at arbejde med i dag, står i en tekstfil. I denne øvelse skal du sørge for, at data kan hentes fra tekstfilen til en liste af Students, og data fra en liste af Students kan skrives til tekstfilen. Denne funktion skal håndteres via din egen DataHandler-klasse, en særskilt C#-klasse du har implementeret sidste uge.

Udfør følgende:

* Tilføj en C#-klasse med navnet DataHandler til projektet.
* Kopi DataHandler klasse fra sidste uge opgave og indsæt den i den tilføjede klasse.
* Tilpasse den til Student klasse. Dvs. Erstat SavePersons og LoadPersons metoder til SaveStudents og LoadStudents som returnerer en liste af Students.

### Øvelse 3.1: Hent data fra tekstfilen

Nu du vil hente data fra tekstfilen. Tekstfilen (students,txt) er lagt sammen med denn opgave på itsLearning.

Udfør følgende:

* I Program.cs benyt DataHandler klasse til at hente studeredenes oplysninger og gem dem i en liste/array.
* Vis studerendes oplysninger på konsolvinduet. (Bonus vis dem i en tabuler måde og hver gruppe i en farve)

**Fremgangsmåde: (for følgende opgaver)**

* Del teamet op i 2 grupper, og brug parprogrammering.
* Team: Del 2 algoritmer (f.eks. selection sort og bubble sort) ud mellem de to grupper.
* Implementé algoritmerne i de to grupper.
* Bliv enige om et tidspunkt for videndeling, hvor de to grupper demonstrer hvad der blev lavet.
* Giv feedback og stille spørgsmål.
* Refaktore koden på baggrund af feedback.

### Øvelse 4: Sortering algoritmer

I denne øvelse arbejder du med sortering algoritmer. Der er indbygget sort metoder med forskellige datastrukturer (arrays, lists, osv.), men formålet med opgaven er at du selv implementerer algoritmer for at du ved logikken bag dem og øger din computational thinking.

Udfør følgende:

* Tilføj en C#-klasse med navnet SortingAlgorithms til projektet.

### Øvelse 4.1: Selection sort algoritm

På baggrund af forberedelses materialer, der blev lagt i asynkrone undervisning til algoritmer, burde du at være klar til at implementere selection sort algoritm, som en metode i SortingAlgorithms klasse.

Udfør følgende:

* Lav pseudokode for selection sort algoritm.
* Implementé SelectionSort metode, der har en parameter som en liste af studerende og den sorterer studerende basereret GruppNumber.

### Øvelse 4.2: Bubble sort algoritm

På baggrund af forberedelses materialer, der blev lagt i asynkrone undervisning til algoritmer, burde du at være klar til at implementere bubble sort algoritm, som en metode i SortingAlgorithms klasse.

Udfør følgende:

* Lav pseudokode for bubble sort algoritm.
* Implementé BubbleSort metode, der har en parameter som en liste af studerende og den sorterer studerende basereret på FullName.

### Øvelse 4.3: Quick sort algoritm

På baggrund af forberedelses materialer, der blev lagt i asynkrone undervisning til algoritmer, burde du at være klar til at implementere quick sort algoritm, som en metode i SortingAlgorithms klasse.

Udfør følgende:

* Lav pseudokode for quick sort algoritm.
* Implementé Partition metode, der har 3 parametre. En parameter som en liste af studerende og en int low og en anden int high. Metoden deler listen i to mindre liste.
* Implementé QuickSort metode, der har de samme 3 angivet parametre i partition metode. Metoden sorterer studerende basereret på FullName.

### Øvelse 4.4: Skriv sorteret data i en tekstfil

Nu efter du har implementeret sortering algoritmer skal du benytte dem i dit program.

Udfør følgende:

* I Program.cs skal du vise en menu for brugeren for at vælge sortering på Group Number eller på Navn og gemme valg i en variabel.
* Hvis valg er på Group Number, indkald selection sort metode.
* Hvis valg er på Navn, vis en ny menu for brugeren for at vælge Bubble sort eller Quick sort og indkald den tilsvarende metode.
* På baggrund af valget, skal du nu benytte DataHandler klasse for at skrive sorteret data i en separat tekstfil.
* Vis en besked til brugeren, hvis skrivning er lykkedes, ellers vis en fejl besked.

### Øvelse 5: Søgning algoritmer

I denne øvelse arbejder du med søgning algoritmer. Der er indbygget søgning metoder med forskellige datastrukturer (arrays, lists, osv.), men formålet med opgaven er at du selv implementerer algoritmer for at du ved logikken bag dem og øger din computational thinking.

Udfør følgende:

* Tilføj en C#-klasse med navnet SearchAlgorithms til projektet.

### Øvelse 5.1: Lineær søgning algoritm

På baggrund af forberedelses materialer, der blev lagt i asynkrone undervisning til algoritmer, burde du at være klar til at implementere lineær algoritm, som en metode i SearchAlgorithms klasse.

Udfør følgende:

* Lav pseudokode for lineær søgning algoritm.
* Implementé LinearSearch metode, der har en parameter som en liste af studerende og fullName som en string. Metoden vil returnere denne studerende med angivet fullName, hvis den er findes i listen. Ellers returnerer den null.

### Øvelse 5.2: Binær søgning algoritm

På baggrund af forberedelses materialer, der blev lagt i asynkrone undervisning til algoritmer, burde du at være klar til at implementere binær søgning algoritm, som en metode i SearchAlgorithms klasse.

Udfør følgende:

* Lav pseudokode for binær søgning algoritm.
* Implementé BinarySearch metode, der har en parameter som en liste af studerende og fullName som en string. Metoden vil returnere denne studerende med angivet fullName, hvis den er findes i listen. Ellers returnerer den null.

### Øvelse 5.3: Benyt søgning algoritmer i dit program

Nu efter du har implementeret søgning algoritmer skal du benytte dem i dit program.

Udfør følgende:

* I Program.cs skal du tilpasse menuen, du har implementeret for at vise 2 valg Sorting eller Search.
* Så laver du 2 under menuer. Under Sorting laver du en menu, der viser sortering på Group Number eller på Navn. Under Search laver du en menu, der viser Linear eller Binary search.
* Indkald den tilsvarende metode og vis resultatet.
* **Bonus opgave:** vis hvor lang tid tager det til at udføre de forskellige algoritmer og sammenlign dem og diskutere hvorfor.

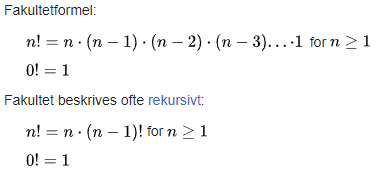
### Øvelse 6: Rekursion

Rekursion er en metode, der kalder sig selv. I denne øvelse arbejder du med rekursion. Binær søgning algoritm i øvelse 5.2 kan godt laves om til at du kan benytte rekursion.

Udfør følgende:

* Implementé en ny BinarySearch metode og kalde den RecursiveBinarySearch i SearchAlgorithms klasse.
* Benyt den i dit Program.cs.

### Øvelse 6.1: Beregn fakultet for et tal ved hjælp af rekursion



Udfør følgende:

* Opret et nyt projekt med navn Recursion.
* Implementé Factorial metode, der afregner fakultetstal for et helttal.
* Benyt metoden i dit program.

### Øvelse 6.2: Tower of Hanoi

Følg artiklen om [Program for Tower of Hanoi Algorithm](https://www.geeksforgeeks.org/c-program-for-tower-of-hanoi/?ref=lbp) og udvikl et program der viser hvordan Tower of Hanoi problemstilling bliver løst ved hjælp af rekursion.

### Øvelse 7: Review-spørgsmål

* Forklar kort hvordan de tre sorteringsalgoritmer (BubbleSort, SelectionSort og QuickSort) fungerer.
* Hvorfor er det vigtigt at sortere listen, inden der udføres en binær søgning?
* Hvordan adskiller lineær søgning sig fra binær søgning, og hvornår er hver metode mest hensigtsmæssig?
* Hvad kræver rekursion, og hvordan adskiller den sig fra iterative løkker i programmering?

**Vi mødes til videndeling kl. 14:30 i generel.**