Programmering C | Afsluttende projekt

- Projektbeskrivelse -

```html
<!DOCTYPE html>
<html>
<head>
  <script src="script.js">
  </script>
</head>
<body>
  <input type="text" id="indput">
  <button onclick="yeet()"> Klik her for konvertering </button>
  <p id="text-box"> yeet </p>
  <p id="result"> Result </p>
</body>
</html>
```

```javascript
console.log("Hello there!");

convert = true;

function conversion(inputValue){
  var short = document.getElementById('result');
  if (convert == true){
    console.log("yeet");
    if (inputValue === "A"){
      console.log(10);
      short.innerHTML = "Resultatet er 10"
      convert = false;
      console.log(convert);
    }
    else if (inputValue === "B"){
      console.log(11);

    }
    else if (inputValue === "C") {
      console.log(12);
    }
    else if (inputValue === "D") {
      console.log(13);
    }
    else if (inputValue === "E") {
      console.log(14);
    }
    else if (inputValue === "F") {
      console.log(15);
    }
    else {
      console.log("nAn");

    }
  }
}

function yeet(){
  var inputValue = document.getElementById('indput').value;
  document.getElementById("text-box").innerHTML =
(document.getElementById('indput').value);
  if (convert == true){
    conversion(inputValue);
    console.log("true");
  }
  else {
    console.log("stop");
  }
}
```

Elever: Ferhat Akdeniz & Nathaniel Finn Michel Risum

Vejleder/lærer: Karl G Bjarnason (kgb)

Afleverings dato: 04.03.2019

Skole: HTX - Roskilde

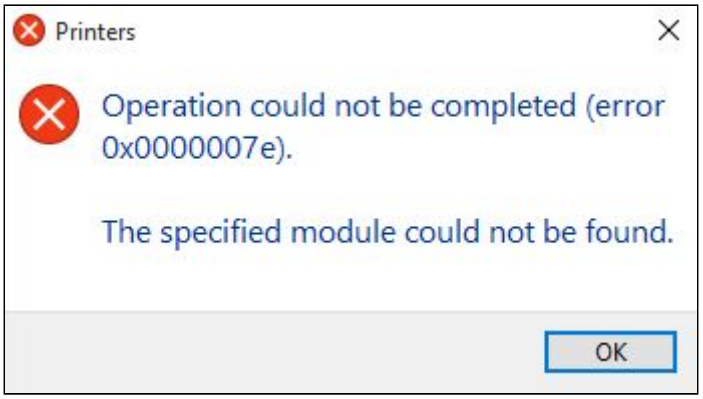
Klasse: 3.4 Mat/IT

Fag: Programmering C

Indholdsfortegnelse

1. Analyse

Inden for verdenen af programmering kan man støde på fejlkoder. Microsoft er kendte for deres såkaldte BSoD - Blue Screen of Death - i Windows. Når man støder på denne, bliver man rigtigt ofte forsynet med en fejlkode, eksempelvis som det ses i billedet herunder.



For at bruge disse fejlkoder til noget, skal man have en vis interesse for fejlfinding og altså højst sandsynligt også en interesse for programmering. Man skal også have en vis kendskab til hexadecimale tal, hvis man hurtigt skal kunne aflæse koden for derefter at slå denne op i et datablad.

1.1. Kerneproblemer

Her kan en række kerneproblemer på baggrund af vores analyse ses:

1. Hvordan laver vi konvertering imellem de heksadecimale og decimale talsystemer.
2. Hvordan skal brugeren kunne lave input på websiden?
3. Hvordan skal konverteringen skelne mellem tal og bogstaver?
4. Brugeren skal finde ud af, hvordan det tal konvertering fungerer.

På baggrund af ovenstående kerneproblemer har vi opstillet en problemformulering, som kan ses i næste kapitel.

2. Definition af målgruppe

I følgende projekt vil vi gerne appellere til de interesserede IT unger. Vi vil derfor tage hensyn til de unge mennesker, som har en kæmpe stor interesse inden for programmering og selve udviklingen af koden. Sidst men ikke mindst selve systemet af IT systemer, herunder hvordan et simpelt program af konvertering af tal kan laves samt dokumenteres. Dette kræver dog, at de enkelte individ mere eller mindre har en interesse inden for emnet.

Derfor er vores målgruppe altså ikke nødvendigvis unge, men personer med nogen forståelse for matematik og systematik og skelnen mellem talsystemer.

2.1. User Stories

Nedenstående ses 3 eksempler på 3 forskellige User Stories. En User Story er en kort og simpel beskrivelse af opnåelse af et problem. User Stories skrives som regel ud fra en brugers synspunkt, som mere eller mindre ønsker en eller anden ny funktionalitet fra et givent IT-system.

Herunder taler man om følgende 3 områder:

1. Hvem - som "rolle"
2. Hvad - ønsker jeg at "ønske"
3. Hvorfor/fordi - med det formål at "formål"

2.1.1. User Story 1

1. Hvem - Jeg er en elev på HTX-Roskilde.
2. Hvad - Som gerne vil lære, om hvordan systemudvikling fungerer på en simpel men informativ måde.
3. Hvorfor/fordi - Fordi jeg gerne vil bruge det i min nærmere fremtid.

2.1.2. User Story 2

1. Hvem - Jeg er en lærer på HTX-Roskilde.
2. Hvad - Som gerne vil lære mine elever om udvikling af et bruger interaktivt system i Javascript.
3. Hvorfor/fordi - Fordi jeg gerne vil bruge det i min nærmere fremtid.

2.1.3. User Story 3

1. Hvem - Jeg er elev på Roskilde Gymnasium.
2. Hvad - Som gerne vil lære om kode udviklingen bag tal konvertering.
3. Hvorfor/fordi - Fordi jeg gerne vil programmere i Javascript.

3. Problemformulering

Nedenstående ses vores foreløbig problemformulering:

> Fejlkoder angives i mange systemer som hexadecimale tal, og disse kan være svære at tyde. Kan vi lave et værktøj til at konvertere disse tal til decimal/binært og tilbage igen?

4. Løsningsforslag (3 stk)

Nedenstående ses vores 3 foreløbige løsningsforslag til projektet:

1. Konvertering med Python
2. Konvertering med JavaScript
3. Bygge en maskine med Arduino

4.1. Diskussion af valg af løsningsforslag

Måden hvorpå denne konvertering skal foregå er egentlig, at brugeren skal få en kort introduktion til omregningen mellem decimal, binære og hexadecimale tal og de vil via input kunne taste værdier og ved tryk på en knap, sker konvertering altså. Derfor vælger vi konverteringen med JavaScript, fordi Javascript har den fordel, at der er tale om nem kompatibilitet når vi har det i HTML, eksempelvis vores studieweb/website.

4.2. Udvikling af spike solutions

Den første spike solution der udvikles i dette projekt har fokus på konvertering mellem et hexadecimalt tal til et decimal. Tallet i dette tilfælde er blot på et ciffer, og kommer ikke til at blive noget helt specielt.

Det første vi starter med er at lave en ren HTML-fil. I denne kommer der til at være et link til et script som egentlig kommer til at indeholde kernefunktionaliteten i denne spikesolution. Ydermere skal der være et input felt hvor brugeren kan taste sin hexadecimale værdi, en knap til at starte konverteringen, et felt til at vise input og et til at vise resultatet.

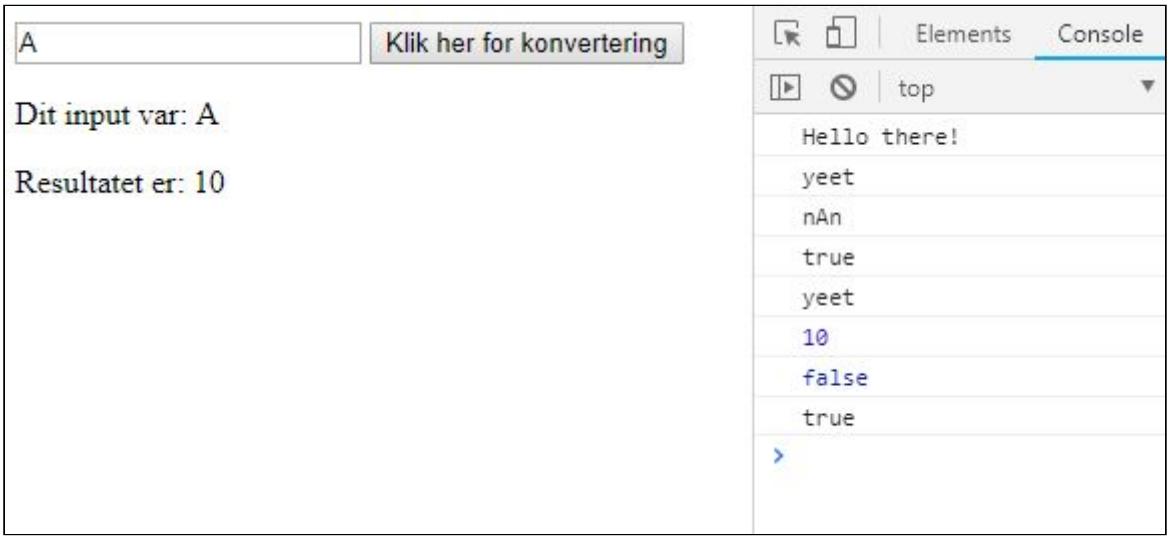
Den fulde html-fil ses i afsnit 7 - metakode , - Herunder ses dog de relevante udsnit af filen.

```
<input type="text" id="indput">
  <button onclick="yeet()"> Klik her for konvertering </button>
  <p id="text-box"> yeet </p>
  <p id="result"> Result </p>
```

Den anden fil der oprettes for denne spikesolution er en JavaScript-fil -- script.js

I denne vil der først og fremmest printes "Hello there!" til konsollen. Derefter sættes boolean-værdien af convert til 'true'. 'convert' er nemlig et udtryk vi bruger flere steder i vores spike-solution til at se, om der er foretaget en konvertering endnu. Som det næste defineres konverteringsfunktionen, som tager et parameter som vi selv har defineret - inputValue. Denne inputValue er værdien af det inputfelt brugeren har at skrive i, men vil først blive læst når der klikkes på knappen. Derefter tjekkes der om 'convert' er true. Hvis den er, vil værdien af input sammenlignes med 16 forskellige løkker.
Nu da koden er så kort som den er, vil vi ikke endnu tilknytte kommentarer til filerne

Herunder ses input af A. Den decimale værdi for A er 10, så hvad der ses i testen passer nu meget godt.



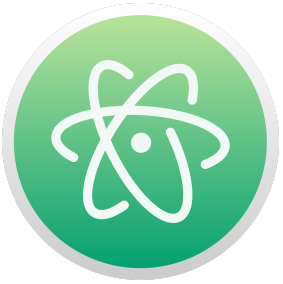
5. Valg af værktøjer

I følgende projekt benytter vi 2 teksteditor, herunder Atom og Brackets, som begge er open source programmer.

5.1. Brackets

Brackets er et open-source samt cross-platform editing program, hvilket betyder både OS X, Windows og Linux har fuldt adgang til denne tekst editor. Brackets er til gengæld et værktøj, som har fokus på de visuelle værktøjer og præprocessor support. Bracket er et moderne tekstredigeringsprogram, der gør det nemt at designe i browseren. Det er egnet for web designers og front-end udvikler.

5.2. Atom



Atom er en open-source og cross platform text-editor som kører i sin egentlighed består af HTML, JavaScript, CSS og Node.js integrering. Det kører på Electron, et framework til at lave applikationer på tværs af platforme med web-teknologier. Med i Atom er integration af GitHub, så man altså let kan både 'push' og 'pull' kode i realtid med andre brugere. Der er også en indbygget package-manager, så man nemt kan installere og fjerne moduler for at ændre sin oplevelse af Atom, hvilket essentielt giver en frihed der minder om den samme frihed man har i Linux.

Atom har den fordel, som hjælper brugeren i at skrive en kode hurtigere med en smart og fleksibel autofuldførelse. Atom kan opdele din Atom-grænseflade i flere ruder for at sammenligne og redigere kode på tværs af filer og giver også mulighed for at lede efter filer eller søgeord i ens filsystem.

5.3. License

5.3.1. Brackets

Brackets er udgivet under MIT-licensen som er vedhæftet herunder:

```
Copyright (c) 2012 - present Adobe Systems Incorporated. All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a
copy of this software and associated documentation files (the "Software"),
to deal in the Software without restriction, including without limitation
the rights to use, copy, modify, merge, publish, distribute, sublicense,
and/or sell copies of the Software, and to permit persons to whom the
Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
DEALINGS IN THE SOFTWARE.
```

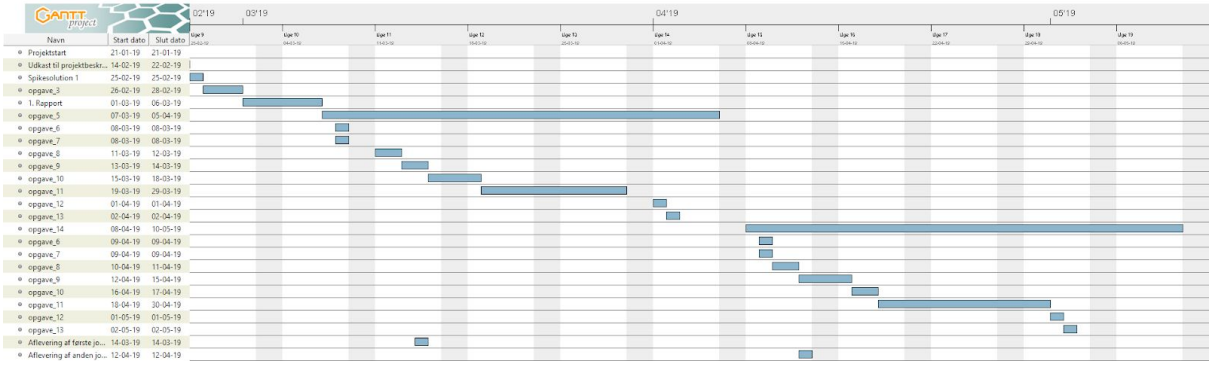
5.3.2. Atom

Atom er også open-source og kan hentes via deres hjemmeside https://atom.io/ eller via deres repository på GitHub https://github.com/atom/atom

6. Tidsplan, inkl. ansvarsfordeling

Der er brugt GanttProject til at lave en tidsplan, som er meget brugervenligt og overskueligt at bruge i arbejdsprojekter.

Tidsplanen kan ses nedenstående:



I tidsplanen ses tydeligt, hvordan vi har to iterationer af udviklingsforløbet med planlægning, design, implementering test og repetition.

7. Metakode

Nedenstående kan vores kode af programmet ses:

HTML:

```html
<!DOCTYPE html>
<html>
<head>
  <script src="script.js">
  </script>
</head>
<body>
  <input type="text" id="indput">
  <button onclick="check()"> Klik her for konvertering </button>
  <p id="text-box"> yeet </p>
  <p id="result"> Result </p>
</body>
</html>
```

JavaScript:

```javascript
console.log("Hello there!");

convert = true;

function conversion(inputValue){
  var short = document.getElementById('result');
  if (convert == true){
    console.log("yeet");
    if (inputValue === "A"){
      console.log(10);
      short.innerHTML = "Resultatet er: 10"
      convert = false;
      console.log(convert);
    }
    else if (inputValue === "1") {
      console.log(1);
    }
    else if (inputValue === "2") {
      console.log(2);
    }
    else if (inputValue === "3") {
      console.log(3);
    }
    else if (inputValue === "4") {
      console.log(4);
    }
    else if (inputValue === "5") {
      console.log(5);
    }
    else if (inputValue === "6") {
      console.log(6);
```

```
    }
    else if (inputValue === "7") {
      console.log(7);
    }
    else if (inputValue === "8") {
      console.log(8);
    }
    else if (inputValue === "9") {
      console.log(9);
    }
    else if (inputValue === "B"){
      console.log(11);

    }
    else if (inputValue === "C") {
      console.log(12);
    }
    else if (inputValue === "D") {
      console.log(13);
    }
    else if (inputValue === "E") {
      console.log(14);
    }
    else if (inputValue === "F") {
      console.log(15);
    }
    else {
      console.log("nAn");

    }
  }
}

function yeet(){
  var inputValue = document.getElementById('indput').value;
  document.getElementById("text-box").innerHTML = "Dit input var:
"+(document.getElementById('indput').value);
  if (convert == true){
    conversion(inputValue);
    console.log("true");
  }
  else {
    console.log("stop");
  }
}

```