# Map Generator

## Allmänt

Ni kommer i den här uppgiften att få skapa en map generator (hädanefter MG) som ni kan använda i spel eller liknande.

MG kommer att ha funktionalitet för att kunna generera enskilda rum med olika karaktäristik samt ha funktioner för att navigera mellan olika rum.

I uppgiften kommer ni till en början att skapa en text-mässig beskrivning av er karta, men denna kommer mot slutet att lyftas in i en grafisk miljö där vi beskriver den visuellt.

## Syfte

Att på ett lekfullt och tydligt sätt:

* Skapa och fördjupa kunskaper om hur man delar upp större projekt i mindre hanterbara delar (divide et impera)
* Introducera algoritmer för att lösa problem
* Introducera abstrakta datatyper
* Introducera DOM
* Introducera enkel AI

## Bakgrund

Uppgiften är (i mitt tycke) kul då den har en stegrande svårighetsnivå som gör den lätt att börja med men svårare mot slutet. Den blir dock aldrig supersvår.

Den är också lätt att relatera till då vi kan ta exempel från allt som omger oss, även spel.

Tänk er följande lilla by

I byn ser vi visuellt hur allt hänger ihop. Vi förstår att det bor sex personer här med olika korta och abnorma namn.

Tänk er nu att vi dricker kaffe/te/öl/vatten hos Ada och vill hälsa på Fi också. Ni knallar ut genom dörren och skall bestämma väg. Ni inser ju (rent grafiskt) att det bästa är att gå förbi Eh först och sedan svänga av mot Fi. Men, det är ju fullt möjligt att också knalla förbi Bo, Cid, Doh och sedan Eh och till slut Fi. Det är en uppenbart längre väg, men fullt möjlig.

Det vårt projekt skall mynna ut i är två saker (egentligen fler) och den första är att vi skall skapa en program som kan skapa sådana här kartor. Den andra saken är att vi skall skapa en funktion som kan ta fram alla vägar mellan två punkter på en sådan här karta, oavsett hur komplicerad den än är.

## Arbetsformer

Det som ger absolut mest är naturligtvis om man jobbar ensam med en sådan här uppgift, men ni kommer många gånger behöva stöd och hjälp för uppgiften blir tjurig med tiden.

Jag tänker mig att ni skall ha en peer som rubberduck (pratkompis) som ni kan ta hjälp av och jag vill att ni vänder er till honom/henne i första hand för att inte någon eller några av er skall bli överöst med frågor 😊

## Deluppgifter

Då hela uppgiften är ganska stor och kommer omfatta åtminstone 500+ rader med kod tänkte jag att jag hjälper er genom att dela upp uppgiften i följande delar.

### NameGen()

**string NameGen() { … }**

Skapa en funktion som kan generera rumsnamn. Rumsnamnen är till för att ge karaktär åt er karta genom unika namn. Exempel på rum som ni kan skapa är

Glorious Hallway of Illustrious Frogs

Sinister Alcove of Smelly Feet

Som du kan se byggs namnen upp av adjektiv, adverb och substantiv på ett enkelt sätt. Sedan slumpar man vilket av de olika orden man vill ha från varje klass och slår ihop dem. Returnera strängen och funktionen är klar.

### Definiera class Location

Du kommer att behöva en klass för att hantera ett abstrakt objekt kallat Location. Location innehåller en unik identitet för just den här platsen, ett namn, en miljö samt en lista med exits (vilka andra rum det ansluter till).

### RoomGen()

**Room RoomGen( ) { … }**

Den här funktionen skall skapa ett objekt av typen Room (egen klass). Från funktionen skall NameGen anropas för att ge rummet ett unikt namn. En miljö bör skapas utifrån en lista av olika typer av miljöer (ex ”Stone”, ”Grass”, ”Gravel”, ”Dirt” m.m.)

Arrayen med exits skall vara tom.

### MapGen( nr\_of\_rooms, complexity )

Den här funktionen skall skapa en cirkulär karta av sammanhängande rum. Antalet rum styrs av nr\_of\_rooms. Complexity är antalet extra förbindelser som skall skapas slumpmässigt i kartan. Detta ökar alternativa vägar mellan rummen så begränsa complexity < nr\_of\_rooms/2

Funktionen returnerar en array med rum.

### AllPaths(source, destination)

Ok, nu blir det lite svårare och jobbigare….

Den här funktionen skall undersöka alla möjliga vägar mellan source och destination och returnera en lista med med de kortaste vägarna först.

## Lite svårare deluppgift

Miljön rummet befinner sig i skall associeras med en kostnad. Ex Dirt = 2, Stone = 1, Grass = 3 …. Denna kostnad skall tas hänsyn till då path beräknas och AllPaths skall alltid returnera billigaste vägen.