Norges teknisk–naturvitenskapelige universitet
Institutt for datateknikk og informasjonsvitenskap

TDT4102 Prosedyre og Objektorientert programmering Vår 2015

Øving 8

Frist: 2015-03-13

Mål for denne øvinga:

- Selvstending utvikling av klasser
- Repetisjon av dynamisk minnehåndtering
- Installasjon og bruk av tredjeparts bibliotek

Generelle krav:

- Bruk de eksakte navn og spesifikasjoner gitt i oppgaven.
- Det er valgfritt om du vil bruke en IDE (Visual Studio, XCode), men koden må være enkel å lese, kompilere og kjøre.

Anbefalt lesestoff:

- Kapitler 6 og 10, Absolute C++ (Walter Savitch)
- http://www.sfml-dev.org/tutorials/2.2/#getting-started

MERK:

- Når man implementerer klasser er det vanlig å lage seg én .h-fil og én .cpp-fil per klasse (f.eks. hvis klassen heter «Car», lager man Car.cpp og Car.h, og skriver all koden for klassen «Car» i disse filene.) Følg denne normen i oppgaver hvor det bes om å skrive klasser.

I denne øvingen skal du lage spillet Minesveiper. Minesveiper består av et rektangulært spillebrett som er ett rutenett av firkanter. Under et satt antall tilfeldige ruter ligger det miner og spilleren må åpne alle rutene som ikke inneholder miner for å vinne spillet. Hvis spilleren åpner en rute som inneholder en mine, har han tapt. Ruter som ikke inneholder en mine, har et nummer fra null til åtte som representerer hvor mange miner det finnes i naborutene. Siden spilleren skal ha mulighet for å stille vanskelighetsgraden til spillet selv må arrayet eller arrayene som representerer spillebrettet være dynamiske, slik at spilleren kan bestemme størrelsen på brettet.

Siden det å spille minesveiper i konsollen fort blir tungvindt, skal vi installere et bibliotek kalt SFML som lar oss lage et grafisk brukergrensesnitt, og der får du utdelt litt kode som avhenger av at `Minesweeper`-klassens `public`-metoder er implementert presist som skissert i oppgaven for at koden skal virke. Utover det står man egentlig ganske fritt til å implementere spillet som man vil. Oppgavene her skisserer en mulig løsning.

```
class Minesweeper {
private:
    // Legg til de medlemsvariablene og hjelpefunksjonene du trenger

public:
    Minesweeper(int width, int height, int mines);
    ~Minesweeper();

    bool isGameOver() const;

    bool isTileOpen(int row, int col) const;
    bool isTileMine(int row, int col) const;

    void openTile(int row, int col);

    int numAdjacentMines(int row, int col) const;

    // Vi slår av autogenerert kopikonstruktør og tilordningsoperator for å unngå feil
    Minesweeper(const Minesweeper &) = delete;
    Minesweeper &operator=(const Minesweeper &) = delete;
};
```

1 Sette opp prosjektet og installere SFML (25%)

a) Lag et nytt prosjekt og legg til de filene du fikk utdelt. Pakk ut `oving08.zip` og legg til `minesweeper.h`, `minesweeper.cpp` og `main.cpp` i prosjektet ditt. Pass også på at filen `sansation.ttf` ligger i prosjektmappa slik at programmet kan finne den når du kjører det.

b) Installer SFML og følg instruksjonene for å legge det til i prosjektet.
Gå til `http://www.sfml-dev.org/tutorials/2.2/` og følg stegene for din platform (Visual Studio, XCode, osv.)
Legg til følgende biblioteker i prosjektet:

- `sfml-system`
- `sfml-window`
- `sfml-graphics`

2 Datastruktur (35%)

Siden spillet består av et rutenett, vil det være naturlig å bruke en tabell for å representere tilstanden til spillebrettet. For hver rute må spillet holde styr på om ruten har blitt åpnet eller ikke og om den inneholder en mine eller ikke, så man kan for eksempel bruke en `struct` til å representere dette:

```
struct Tile {
    bool open;
    bool mine;
};
```

Når brukeren starter en ny runde, vil det grafiske brukergrensesnittet opprette et `Minesweeper`-objekt. Konstruktøren mottar argumenter som spesifiserer hvor stort brettet skal være og hvor mange miner det skal inneholde. Brukergrensesnittet vil kalle på funksjonene `isTileOpen()`, `isTileMine()` og `numAdjacentMines()` når det skal tegne brettet, og det vil kalle på `openTile()` når spilleren klikker på en rute.

- Når spillet starter skal alle rutene være lukket.
- `isTileOpen()` skal returnere om en rute har blitt åpnet eller ikke.
- `isTileMine()` skal returnere om en rute inneholder en mine eller ikke.
- `openTile()` skal markere en rute som åpnet.

3 Spill-logikk (40%)

Neste steg er å legge til miner og håndtere disse på riktig måte.

- Når spillet starter skal det plasseres et bestemt antall miner på brettet. Antallet er gitt i argumentet `mines` til konstruktøren.

  Merk: Pass på at du ikke plasserer miner i samme rute flere ganger
- Hvis spilleren åpner en rute som inneholder en mine, er spillet over og `isGameOver()` skal returnere `true`.
- `numAdjacentMines()` skal returnere hvor mange av naborutene som inneholder miner.

  Naboskap inkluderer diagonale naboer. Hver rute har altså opptil 8 naboer. Pass på at dette også blir riktig for ruter langs kanten av brettet.

4 Ekstra funksjonalitet (valgfritt)

- Endre `openTile()` slik at hvis ruten ikke inneholder en mine og ingen av naborutene inneholder miner, åpnes også alle naborutene. Gjenta dette rekursivt for naborutene slik at store områder uten miner kan åpnes automatisk.
- Modifiser den utdelte koden og din egen kode som nødvendig for å la brukeren markere miner med flagg ved å høyreklikke på en rute. Hvis man høyreklikker igjen på flagget, skal flagget fjernes.
- Vis hvor mange miner som gjenstår og gjør det mulig å vinne spillet, altså vis en melding dersom alle ruter som ikke inneholder miner har blitt åpnet. Bonus: Gjør dette i det grafiske vinduet, ikke i konsollen.