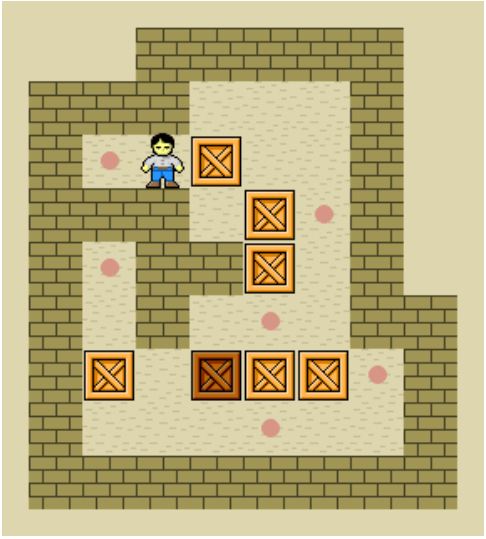
Tællende aktivitet – C++, forår 2021



Denne opgave går ud på at programmere spillet Sokoban. Spillet går ud på at skubbe kasser gennem en labyrint således at de ender på en række mål felter. I kan prøve at spille spillet på https://www.mathsisfun.com/games/sokoban.html .

Reglerne er for spillet er som følger:

1) Et felt defineres som frit hvis det er: a) et tomt felt, eller b) et mål felt. Dette er uafhænging af om der står en kasse eller en spiller på feltet.

2) Spilleren kan bevæge sig ét felt ad gangen i fire retninger (op, ned, højre, venstre) hvis det felt der flyttes til er frit. Dog kan spilleren kun bevæge sig til et felt med en kasse hvis denne kasse kan skubbes til. I så fald bliver kassen skubbet således at spilleren står på kassens tidligere plads og kassen flytter et enkelt felt i bevægelsesretningen.

3) En kasse kan skubbes til hvis næste felt i bevægelsesretningen er frit og ikke indeholder en anden kasse.

Programmet skal skrives efter nedenstående specifikationer. I må gerne tilføje de metoder og variable I finder nødvendige.

Programmet skal indeholde klasserne Pos, Board, Level og Game, som er beskrevet herunder.

Pos: Class eller struct med public membervariable i og j, der repræsenterer en position på boardet. Da denne klasse/struct kun indeholder member 2 variable og ingen metoder kan den med fordel defineres direkte i header filen, dvs. der vil ikke være nogen pos.cpp fil.

Board: Indeholder en enkelt banes tilstand, dvs. information om banen, kassernes positioner og spillerens position. Board holder styr på de statiske elementer vha. et 2D grid af char[1]. Kassernes positioner repræsenteres ved en std::vector eller array af Pos. addBox og remove box returnerer en bool der beskriver om operationen var successful (der må kun være én box på det samme felt og man kan ikke fjerne en box hvis der ikke er en). Constructoren til Board tager in streng som input som beskriver banen (se eksempel nedenfor i opgavebeskrivelsen):

public:

Board(std::string boardString)

char getStaticElement(Pos p)

Pos getPlayerPosition()

void setPlayerPosition(Pos p)

bool addBox(Pos p)

bool removeBox(Pos p)

bool isBox(Pos p)

private:

std::vector<std::vector<char> > staticElements *// I må også godt gemme det som std::vector<char> og selv holde styr på indexeringen i et 2D grid. Alternativt kan I også benytte arrays*

std::vector<Pos> boxes *// I må også godt benytte arrays*

Pos playerPosition

Level: Level holder styr på den logik der bliver brugt til at manipulere boarded når spilleren bevæger sig. Den håndterer altså bevægelsen af spilleren, tjekker at det er lovlige bevægelser, flytter kasserne, visualiserer boardet, og tjekker om kasserne står på målpositionerne. Klassen indholder desuden en reset metode der sætter boarded tilbage til sin oprindelige tilstand. Level har følgende metoder (vær opmærksom på at I skal have defineret en enum (eller enum class) der hedder Direction som kan have værdierne LEFT, RIGHT, UP, DOWN):

Level(Board *board)

bool move(Direction d): Updaterer Board således at spilleren og kasserne har rykket sig som de skal. Returnerer true/false alt efter om bevægelsen var mulig.

void reset(): Genskaber boardet's oprindelige opsætning

std::string visualize(): Returnerer en std::string der er en visualisering af boarded (see boardString nedenfor)

isDone(): Returnerer om spillet er vundet, dvs. om alle kasserne står på målfelterne

Game: Klasse der står for at køre et loop der tager læser brugerens input (gennem std::cin) og baseret på dette input enten: a) bevæger spilleren, b) resetter banen, eller c) lukker programmet. Game har følgende metoder:

---

1 En given char i 2D grided repræsenterer wall="W", goal="x" empty=".", og background="-". Når banen visualiseres eller bliver genereret ud fra en std::string gælder der desuden at player="P", box="o" og box-on-a-goal="O"

Game(Level level)*

void run()

Program test

Programmet starter i main-metoden der indeholder en hard-coded textstreng der repræsenterer banen. Ud fra denne textstreng oprettes der et board som gives til constructoren i level. Det er muligt at programmere en Sokuban løsning direkte gennem kald til level, som nedenfor:

```
int main(){
std::string boardString("--WWWW--\n"
                        "WWW..WWW\n"
                        "W..P...W\n"
                        "WW..o.xW\n"
                        "-WWWWWWW");
Board b(boardString);
Level l(&board);
l.move(Direction::DOWN);
l.move(Direction::RIGHT);
l.move(Direction::RIGHT);
l.move(Direction::RIGHT);
std::cout << l.visualize() << std::endl;
std::cout << l.isDone() << std::endl;
}
```

Desuden skal spillet kunne spilles ved hjælp af bruger input ved at køre

```
int main(){
std::string boardString("--WWWW--\n"
                        "WWW..WWW\n"
                        "W..P...W\n"
                        "WW..o.xW\n"
                        "-WWWWWWW");
Board b(boardString); //I skal programmmere denne
Level l(&b);
Game g(&l);
g.run();
```

```
}
```

Sokoban level 1 er givet herunder:

```
--WWWWW-
WWW...W-
WxPo..W-
WWW.oxW-
WxWWo.W-
W.W.x.WW
Wo.OooxW
W...x..W
WWWWWWWW
```

Test kode

I skal selv stå for at teste jeres program. I bedømmelsen af opgaven vil jeres kode potentielt blive kørt med et stykke test kode I ikke har set før. Det er derfor vigtigt at I kan køre testkoden nedenfor uden modifikationer

```
#include <iostream>
#include <string>
#include "board.h"
#include "level.h"
#include "game.h"

int main(){
    std::string boardString("--WWWW--\n"
                            "WWW..WWW\n"
                            "W..P...W\n"
                            "WW..o.xW\n"
                            "-WWWWWWW");
    Board b(boardString);
    Level l(&board);
    l.move(Direction::DOWN);
    l.move(Direction::RIGHT);
    l.move(Direction::RIGHT);
    l.move(Direction::RIGHT);
    std::cout << l.isDone() << std::endl;
}
```

Aflevering

Opgaven skal løses enkeltvis. I må gerne snakke sammen, men det er *ikke tilladt at kopiere hinandens kode*.

Opgaven er stillet den 7. april 2021. Opgaven skal afleveres på itsLearning senest 25. april 2021 kl. kl. 23:59.

OBS: Dette er en tællende aktivitet, dvs. den tæller med i den samlede vurdering af jeres præstation i kurset. Hvis der er noget I kan se at I ikke er i stand til at løse inden deadline kan I således med fordel slække på kravene for så at kunne løse resten af opgaven. Dette trækker selvfølgelig ned i bedømmelsen, men det er bedre at løse en opgave med slækkede krav end at give helt op.

I skal aflevere:

Én samlet zip-fil med navnet: **ta_efternavn_fornavn.zip** (ungå at bruge æ,ø og å)

Zip-filen skal indeholde:

readme.txt : fil hvor I beskriver om I har løst hele opgaven eller har været nødt til at slække på nogen af kravene

Filerne: main.cpp, game.h, level.h, board.h, pos.h, game.cpp, level.cpp, board.cpp