Eksamensopgave i RD2-SWC

13. august 2019

Alle hjælpemidler, som ikke kræver brug af internet, er tilladt.

Aflevering

Der skal afleveres én PDF-fil med opgavebesvarelsen i C++-delen og én PDF-fil med opgavebesvarelsen i Softwareudviklingsdelen. Desuden skal al kildekode afleveres i en zip-fil. En komplet a*flevering indeholder altså **3 filer***.

Til hver C++ opgave er der en ***test-kode***, som skal ***eksekveres*** og et ***screendump*** skal indsættes i afleveringen. Hvis programmet ikke kan compile, så indsættes i stedet ***fejlmeddelelsen*** fra compileren.

Enhver form for kopier/indsæt (copy/paste) fra tidligere opgaver anses som eksamenssnyd. Testkode fra denne eksamensopgave må dog gerne kopieres ind.

Opgavesættet består af 7 sider, 1 forside, 5 sider med C++ opgaver og 1 side med software-udviklingsopgaven.

Opgave 1 (1-1½ time, 35 point)

I denne opgave skal klassen Ticket implementeres. Klassen Ticket må ikke benytte klassen std::string og skal i stedet benytte char-arrays.

```
class Ticket {
public:
    Ticket();

private:
    int mTheater;
    int mTicketNo;
    double mPrice;
    char mDate[11];
    char mTime[6];
    char* mSeat;
    char* mMovie;
    bool m3D;
};
```

Member variablene repræsenterer sal, billetnr, pris, dato, tidspunkt, sæde, film og om det er 3D-film (i rækkefølge fra ovenstående).

Bemærk, at der i denne opgave skal benyttes to typer af C-style strings som hhv. er et char array med fast størrelse samt to strenge af variabel størrelse (sæde og film).

HINT: Husk at C-style strings er termineret med et null-tegn.

a) Implementer constructor, copy og move constructor

I første del af opgaven skal constructor og destructor implementeres.

Implementer desuden get/set funktionerne til alle member-variablene. Her lægges der vægt på korrekt brug af const samt korrekt brug af char-arrays. Alle get-funktioner skal returnere const-pointere/referencer i de tilfælde, hvor der ikke er tale om simple datatyper. På samme måde skal set-metoderne acceptere const-parametre.

Det er et krav, at set-metoderne for dato og tid laver kontrol af input-formatet. Hvis formatet er ukorrekt skal input afvises. Datoen behøver ikke være gyldig, men skal følge formatet DD-MM-YYYY (altså dag-måned-år). Tidspunktet skal følge formatet HH:MM (altså timer:minutter). For både tid og dato kan alle ascii-karakterer accepteres men bindestreg og kolon skal være på korrekte positioner. Dvs. datoen 1A-13-2040 må gerne accepteres som gyldig. Set-metoderne for sæde og film skal kontrollere, at input-strengen eksisterer.

b) Tilføj operatorer

Tilføj følgende operatorer til klassen

- Copy constructor
- Move constructor
- Copy assignment
- Move assignment

Se test-kode på de næste 2 sider (HUSK screendump):

Test-kode

a)

```
std::cout << "Create tickets" << std::endl;
std::vector<Ticket> tickets(2);

std::cout << "Set ticket information" << std::endl;
tickets[0].setTicketNo(1);
std::cout << "Set date" << std::endl;
tickets[0].setDate("13-08-2019");
std::cout << "Set time" << std::endl;
tickets[0].setTime("09:00");
std::cout << "Set seat" << std::endl;
tickets[0].setSeat("A12");
std::cout << "Set movie" << std::endl;
tickets[0].setMovie("Eksamen");
std::cout << "Set price" << std::endl;
tickets[0].setPrice(100);
std::cout << "Set theater" << std::endl;
tickets[0].setTheater(141);

tickets[1].setTicketNo(2);
tickets[1].setDate("01-09-2019");
tickets[1].setTime("08:15");
tickets[1].setSeat("A14");
tickets[1].setMovie("Studiestart");
tickets[1].setPrice(2000);
tickets[1].setTheater(180);

for(Ticket& t : tickets) {
    std::cout << std::right << std::setw(4) << t.getTicketNo()
              << std::right << std::setw(12) << t.getDate()
              << std::right << std::setw(7) << t.getTime()
              << std::right << std::setw(5) << t.getSeat()
              << " " << std::left << std::setw(40) << t.getMovie()
              << std::right << std::setw(5) << t.getPrice()
              << std::right << std::setw(5) << t.getTheater()
              << std::endl;
}
```

b)

Nedenstående kode kræver test-koden til del a)

```
std::cout << "Create extra ticket" << std::endl;
Ticket t;
t.setTicketNo(3);
t.setDate("14-08-2019");
t.setTime("20:30");
t.setMovie("Fast and Furious - Hobbs and Shaw");
t.setPrice(105);
t.setTheater(1);

std::cout << "Copy and move constructors" << std::endl;
tickets.push_back(t);                          // Copy constructor
tickets.emplace_back(std::move(tickets[0])); // Move constructor

std::cout << "Create an empty ticket" << std::endl;
Ticket t1;
tickets.push_back(t1);
tickets.push_back(t1);

std::cout << "Copy and move assignment" << std::endl;
tickets[4] = t;                            // Copy assignment
tickets[5] = std::move(tickets[1]);      // Move assignment

for(Ticket& t : tickets) {
    std::cout << std::right << std::setw(4) << t.getTicketNo()
              << std::right << std::setw(12) << t.getDate()
              << std::right << std::setw(7) << t.getTime()
              << std::right << std::setw(5) << t.getSeat()
              << " " << std::left << std::setw(40) << t.getMovie()
              << std::right << std::setw(5) << t.getPrice()
              << std::right << std::setw(5) << t.getTheater()
              << std::endl;
}
```

Opgave 2 (½-1 time, 15 point)

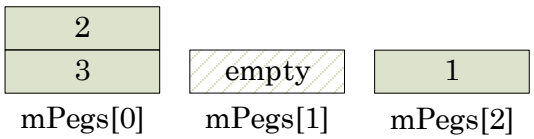
Tower of Hanoi er et velkendt problem. Se evt. bogen side s. 317 for yderligere forklaringer. I denne opgave skal vi IKKE løse algoritmen, da den er givet på forhånd. Vi vil anvende klassen:

```
class TowerHanoi {
public:
    TowerHanoi(unsigned int height = 3);

    void move();
private:
    void move(unsigned int disks, unsigned int source, unsigned int aux,
              unsigned int dest);
    void printTower();

    unsigned int mHeight;
    std::vector<unsigned int> mPegs[3];
};
```

(husk at tilføje includes efter behov)

Member variablene mHeight og mPegs illustrerer hhv. højden af Tower of Hanoi samt de ringe på de forskellige ben/pinde (pegs) hvorpå disse ringe er placeret. Vectoren mPegs[0] repræsenterer altså første ben, mPegs[1] repræsenterer andet ben samt mPegs[2] repræsenterer det tredje ben.

Hvis højden på tårnet er 3, så vil mPegs[0] til at begynde med indeholde data {3,2,1}. Altså her repræsenterer 3-tallet den bredeste ring og 1-tallet er altså den mindste ring. Lad os nu sige, at vi flytter den mindste ring til ben/pind 3, så bliver mPegs[2] data altså {1}, mens mPegs[0] data bliver {3,2}. Grafisk ser det altså således ud:



I denne opgave skal du selv skrive koden til at flytte ringene rundt på de forskellige ben/pinde. Derudover skal du illustrere, hvordan tårnet ser ud i hvert skridt i algoritmen (dvs. implementér funktionen printTower).

Det forventes, at printTower giver et output a'la:

```
       **

    ********

  **********         ****           ******
```

Her er der altså 3 ringe på første ben, 1 ring på andet ben og 1 ring på tredje ben. Som vist herover er bredden af ringene målt i stjerner. Her er mindste ring 2 stjerner, næstmindste 4 stjerner, derefter 6 stjerner osv.

Til hjælp er constructor samt move-funktioner (algoritmen) implementeret og kan ses på næste side. I selve move-funktionen med 4 parametre står der TODO. Her skal koden placeres for at flytte ringene. HINT: funktionerne back(), pop_back() og push_back() kan være nyttige.

```
TowerHanoi::TowerHanoi(unsigned int height) {
    mHeight = height;

    mPegs[0] = std::vector<unsigned int>(mHeight);
    for (unsigned int i = 0; i < mHeight; ++i) {
        mPegs[0][i] = mHeight - i;
    }
}

void TowerHanoi::move() {
    if (mHeight == mPegs[0].size()) {
        move(mHeight, 0, 1, 2);
    } else {
        std::cout << "Error: Tower already moved" << std::endl;
    }
}


void TowerHanoi::move(unsigned int disks, unsigned int source,
                      unsigned int aux, unsigned int dest) {
    if (disks == 1) {
        // TODO: move disk from source to dest

        printTower();
    } else {
        move(disks - 1, source, dest, aux);     // Step 1
        // TODO: move disk from source to dest

        printTower();
        move(disks - 1, aux, source, dest);     // Step 3
    }
}
```

Test

Se test-kode (HUSK screendump):

```
    std::cout << "Tower of Hanoi height 3" << std::endl;
    TowerHanoi th1(3);
    th1.move();

    std::cout << "Tower of Hanoi height 4" << std::endl;
    TowerHanoi th2(4);
    th2.move();
```