Eksamensopgave i RD2-SWC

16. august 2021

Alle hjælpemidler er tilladt. Husk at de generelle regler om snyd stadig gælder.

Aflevering

Der skal afleveres én samlet PDF med opgavebesvarelsen af både C++-delen og softwareudviklingsdelen, som alene er det dokument, der bliver bedømt. I kan kun aflevere et dokument som altså både skal indeholde C++ afleveringen med kildekode og test samt softwareudviklingsdelen.

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. Generelt vurderes opgaverne ud fra en korrekt anvendelse af de tilegnede C++ færdigheder som er opnået gennem gennemførelse af faget. Det forventes, at de tillærte færdigheder anvendes når det kan, herunder anvendelsen af referencer ved ikke simple datatyper samt brug af const, hvor det er muligt.

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

Opgavesættet består af 7 sider, 1 forside, 4 sider med C++ opgaver og 2 sider med software-udviklingsopgaven.

Opgave 1 (1 time, 25 point)

I denne opgave skal vi kigge på en ingrediensliste til en kageopskrift. Når man skal bage en kage, så vil der altså være en liste over ingredienser samt en mængde af de pågældende ingredienser med en enhed.

Til denne opgave skal vi bruge klassen "Ingredient", som repræsenterer én ingrediens. Deklarationen til denne klasse er givet herunder:

```
class Ingredient {
public:
    Ingredient();
    Ingredient(const std::string& name);
    Ingredient(double amount, const std::string& unit, const std::string& name);

    void setAmount(double amount, const std::string& unit);

    double getAmount() const;

private:
    double mAmount;
    std::string mUnit, mName;
};
```

a) Implementer klassen Ingredient

I første del af opgaven skal klassen Ingredient implementeres færdig. Alle funktioner, som skal implementeres, er givet i headeren og forklares herunder

- De tre constructors skal initialisere member variablene ud fra parameterlisten. Husk kun at initialisere de nødvendige variable i de forskellige constructors. Det forventes, at der bruges initializer-lists til alle constructors eller der bruges delegating constructors.
- Membervariablen mAmount angiver mængdeangivelsen af ingrediensen. mName angiver navnet på ingrediensen og mUnit repræsenterer enheden.
- Funktionerne setAmount() og getAmount() hhv. sætter både mængde og enhed på ingrediensen, hvor getAmount() kun er en get-funktion på selve mængden, men ikke enheden. Implementer disse funktioner.
- Implementer getName() og setName() som get/set for mName member. Derudover skal getUnit() implementeres for at kunne få enheden.

b) Implementer insertion operator for Ingredient

Det skal være muligt at bruge insertion operator for Ingredient objekter. Derfor skal insertion operator (<<) implementeres for Ingredient klassen. Operatoren SKAL være implementeret som en friend til klassen. Formatet for udskriften følge nedenstående

- Mængdeangivelsen skal anvende 10 tegn og være højrejusteret
- Mellemrum indsættes
- Enhed skal anvende 5 tegn og være venstrejusteret
- Mellemrum indsættes
- Navn skal være venstrejusteret

c) Multiplikationsoperator for Ingredient

Ingredienserne i test-koden passer til 8-12 banan muffins (mmmh), men man bager jo ofte dobbelt eller trippel-portion – ikke? Så derfor skal vi jo have mulighed for at øge mængden af ingredienser, hvis der skal laves ekstra. Derfor skal multiplikationsoperatoren (operator*) implementeres på klassen, så vi f.eks. kan skrive: `ingredient*2`, hvorefter mængden forøges med en faktor 2.

Test-kode (HUSK Screendump af test)

```
    std::cout << "---------------" << std::endl;
    std::cout << "-- Opgave 1a --" << std::endl;
    std::cout << "---------------" << std::endl;
    std::vector<Ingredient> ingredients;
    ingredients.emplace_back(3, "stk", "\x91g");
    ingredients.emplace_back(170, "g", "sukker");
    ingredients.emplace_back(3, "tsk", "vaniljesukker");
    ingredients.emplace_back(125, "g", "hvedemel");
    ingredients.emplace_back(1, "tsk", "bagepulver");
    ingredients.emplace_back(100, "g", "smeltet sm\x9br");
    ingredients.emplace_back(1, "stk", "stor banan");
    ingredients.emplace_back(100, "g", "hakket m\x9brk chokolade");
    ingredients.emplace_back(8, "stk", "muffinsforme");


    std::cout << "Ingredients : " << std::endl;
    for (Ingredient& i : ingredients) {
        std::cout << i.getAmount() << " " << i.getUnit() << " " << i.getName() <<
std::endl;
    }

    Ingredient aTest;
    aTest.setName("A");
    Ingredient bTest("B");
    std::cout << std::endl;
    std::cout << aTest.getName() << " " << aTest.getAmount() << std::endl;
    std::cout << bTest.getName() << " " << bTest.getAmount() << std::endl;


    std::cout << std::endl << std::endl;
    std::cout << "---------------" << std::endl;
    std::cout << "-- Opgave 1b --" << std::endl;
    std::cout << "---------------" << std::endl;
    // Print ingredient list
    for (Ingredient& ingredient : ingredients) {
        std::cout << ingredient << std::endl;
    }


    std::cout << std::endl << std::endl;
    std::cout << "---------------" << std::endl;
    std::cout << "-- Opgave 1c --" << std::endl;
    std::cout << "---------------" << std::endl;
    // Print ingredient list for twice the amount of muffins
    for (Ingredient& ingredient : ingredients) {
        std::cout << ingredient*2 << std::endl;
    }
    std::cout << std::endl;
    std::cout << "Original" << std::endl;
    // Print ingredient list for twice the amount of muffins
    for (Ingredient& ingredient : ingredients) {
        std::cout << ingredient << std::endl;
    }
```

Opgave 2 (1 time, 25 point)

En opskrift består derudover af en række skridt og nogle instruktioner til hvert skridt. Vi kalder her hvert skridt af instruktioner for en ”paragraph”, som repræsenteres ved klassen:

```
class Paragraph {
public:
    Paragraph(const std::string& str = "") : mStr(str) {}
    const std::string& getText() const { return mStr; }

private:
    std::string mStr;
};
```

Denne klasse er fuldt implementeret i ovenstående og skal blot benyttes herunder. Dertil skal vi have klassen Recipe, der er deklareret som nedenstående:

```
class Recipe
{
public:
    Recipe(const std::string& title);

private:
    std::string mTitle;
    std::vector<Paragraph> mParagraphs;
};
```

Følgende skal implementeres på klassen Recipe

- Constructor skal ændres, så der er en standard titel (default value) på constructorens parameter
- mParagraphs repræsenterer altså alle skridt
- Der skal implementeres en funktion getParagraphs(), som skal returnere en const reference til membervariablen mParagraphs.
- Funktionerne setTitle() og getTitle() skal implementeres for hhv. at sætte og hente membervariablen mTitle.
- Extraction operatoren (operator>>) skal implementeres som friend til klassen. Følgende regler gælder for instruktionerne, der indlæses fra en input stream (istream):
    - Alle linjeskift skal bevares med mindre, der er tale om et nyt skridt (altså ”paragraph”)
    - Når der er to blanke linjer, skal der oprettes et nyt skridt (altså et objekt af typen Paragraph)
        - De to blanke linjer skal IKKE gemmes her
    - Alle skridt gemmes i membervariablen mParagraphs.

Test-koden findes på næste side.

Test

Se test-kode herunder (HUSK screendump):

```
    std::cout << std::endl << std::endl;
    std::cout << "--------------" << std::endl;
    std::cout << "-- Opgave 2 --" << std::endl;
    std::cout << "--------------" << std::endl;

    // Opgave 2
    std::string instructions = "Tag en sk\x86l og sl\x86 \x91ggene ud. \n\nTils\x91t
vaniljesukker og pisk til det bliver helt hvidt og skummende. \n\nDet kan sagtens
tage 5 minutter med en h\x86ndmixer.";
    instructions += "\n\n\n";
    instructions += "S\x86 tils\x91ttes mel og bagepulver.\nNu skal det ikke piskes,
men vendes med en dejskraber.";
    instructions += "\n\n\n";
    instructions += "Sm\x9brret smeltes i en lille gryde og n\x86r det er k\x9blet
af, skal det vendes i dejen.";
    instructions += "\n\n\n";
    instructions += "Bananerne moses i en tallerken.";
    instructions += "\n\n\n";
    instructions += "Chokolade hakkes groft.";
    instructions += "\n\n\n";
    instructions += "Vend chokolade og bananmos i dejen.";
    instructions += "\n\n\n";
    instructions += "Muffinsformene fyldes med dejen (husk de h\x91ver).\nMuffins
bages i 15min ved 170grader forvarmet ovn";
    std::stringstream sInst(instructions);

    Recipe r("Muffins 8 stk");
    sInst >> r;

    std::cout << "   " << r.getTitle() << "   " << std::endl;
    r.setTitle("Muffins 12 stk");
    std::cout << "   " << r.getTitle() << "   " << std::endl << std::endl;
    std::vector<Paragraph> instList = r.getParagraphs();
    for (unsigned int i = 0; i < instList.size(); ++i) {
        std::cout << "Step " << std::setw(3) << i+1 << std::endl;
        std::cout << "--------" << std::endl;
        std::cout << instList[i].getText() << std::endl;
        std::cout << std::endl << std::endl;
    }
```