Exercises

Preben Hagh Strunge Holm, prebenh@mmmi.sdu.dk

March 6, 2022

1

2

3

3.1 Project

Write a class that can calculate the clothing sizes of a person given the height, weight, and age of a person. The class should have these informations as member variables.

The hat size can be calculated by

$$hat = \frac{m}{h} \cdot 41.25 \cdot \pi cm$$

The jacket size is

$$jacket = \frac{h \cdot m}{335} inches + 1/8 \text{ of an inch for each 10 years over age 30}$$

The waist size is

$$waist = \frac{m}{2.6} inches + 1/10 \text{ of an inch for each 2 years over age 28}$$

where m is the weight and h is the height of the person

All inputs are given in centimeters and kg. Note that the age adjustments only take place after 10 full years (or 2 full years). This means, that there is no adjustments during the first years over age 30 or 28.

The final program should allow the user to input the person data, and to choose which size to calculate. The user should have the option to enter new data for a new person or exit the program after each calculation.

3.2 Project

Write a class called QuadraticEquation which corresponds to the equation

$$ax^2 + bx + c = 0$$

One way to do this is to create a class with three member variables of type double representing the constants a, b and c. Create a member function called solve, which solves the equation. Your solution should handle complex numbers as well. One way to do this is to create two functions, one which returns the real part of the solution, and the other returns the imaginary part of the solution.

The main part of the program should allow the user to enter the constants a, b and c. The user should be able to repeat task as many times as needed.

3.3 Project

The Harris–Benedict equations revised by Mifflin and St Jeor estimates the BMR (Basal Metabolic Rate) of a person. The equation is given by:

$$BMR_{men} = 10 \times \text{weight in kg} + (6.25 \times \text{height in cm} - (5 \times \text{age in years}) + 5$$
$$BMR_{women} = 10 \times \text{weight in kg} + (6.25 \times \text{height in cm} - (5 \times \text{age in years}) - 161$$

Create a class that accepts to store information about the gender, weight, height and date of birth. A function to calculate the BMR using the modified Harris Benedict equation must be created.

3.4 Project

The energy consumed when doing exercise can be calculated using the intensity of the workout, the persons body weight and the amount of time spent in the different intensity zones. The intensity is measured by a number called MET (Metabolic Equivalent).

A short list of intensity values

MET	Description
1.3	Sitting quietly and watching TV
2.0	Playing the flute, sitting down
2.5	Playing the violin, sitting down
3.5	Fishing
4.3	Walking 5.6 km/h (3.5 mph)
5.3	Walking uphill 1-5% grade at 4.7 to 5.6 km/h (2.9-3.5 mph)
5.5	Mowing lawn
5.5	Ballroom dancing, fast
8.0	Bicycling moderate 19 to 22 km/h (12-13.9 mph)
9.8	Running 9.7km/h (6mph)
9.8	Swimming, freestyle, fast
11.8	Running 12 km/h (7.5mph)
12.8	Running 14.5 km/h (9 mph)

A more thorough list of MET values for different activities can be found on `https://sites.google.com/site/compendiumofphysicalactivities/Activity-Categories`

The amount of calories burnt can then be calculated by

$$MET \cdot m \cdot t$$

where m is the weight of the person and t is the time spent in hours.

Write a program that allows the user to enter the weight, the type of activity from the above list or directly type the MET value and the time spent in that intensity. The user should be able to enter several activities and get the total amount of calories burnt. Allow the user to enter up to 20 activities and calculate the total calories. After entering the activities, the user should be presented with a list of each activity, the time spent on that activity and the number of calories burnt on that specific activity. The program should use a class to represent the activities and an array (or several) to store the activities.

3.5 Project

Newtons law of universal gravitation between two point masses is given as

$$F = G\frac{m_1 m_2}{r^2}$$

where F is the force between the masses, m_1 and m_2 are the two masses, r is the distance between, and $G = 6.674 \cdot 10^{-11} N \cdot (m/kg)^2$ is the gravitational constant.

Write a class to represent a mass. The class must have a member function

```
gravitation(Mass m, double distance)
```

to calculate the force of attraction. The function must have parameters m and distance, representing the mass of the other object and the distance between.

3.6 Project

Write a program that tests if a number is also a palindrome. A palindrome is a number or text phrase that reads the same backward as forward. For example, each of the following five-digit integers is a palindrome 12321, 55555, 45554, and 11611. Write a program that reads a 5 digit integer and determines whether it's a palindrome (HINT: Use the division and remainder operator to separate the number into its individuel digits).

4

4.1 Project

The Fibonacci numbers F_n are defined as follows. F_0 is 1, F_1 is 1, and

$$F_{i+2} = F_i + F_{i+1}$$

where $i = 0, 1, 2, \ldots$.

In other words, each number is the sum of the previous two numbers. The first few Fibonacci numbers are 1, 1, 2, 3, 5, and 8. One place that these numbers occur is as certain population growth rates. If a population has no deaths, then the series shows the size of the population after each time period. It takes an organism two time periods to mature to reproducing age, and then the organism reproduces once every time period. The formula applies most straightforwardly to asexual reproduction at a rate of one offspring per time period. Assume that the green crud population grows at this rate and has a time period of 5 days. Hence, if a green crud population starts out as 10 pounds of crud, then in 5 days there is still 10 pounds of crud; in 10 days there is 20 pounds of crud, in 15 days 30 pounds, in 20 days 50 pounds, and so forth. Write a program that takes both the initial size of a green crud population (in pounds) and a number of days as input, and that outputs the number of pounds of green crud after that many days. Assume that the population size is the same for 4 days and then increases every fifth day. Your program should allow the user to repeat this calculation as often as desired.

Use a while loop to genereate the fibonacci numbers. Try also to make an infinite loop (while (true) // code) and see what happens.

4.2 Project

The Monty Hall Game Show problem is a famous statistical problem. Behind one out of three doors, there is a big prize (originally a brand new car). At first, the player is asked to select a door. Then the game host presents one of the other doors to the player, behind which no prize was found. The player now have the option to change the door or keep the original door chosen.

Write a program that simulates 1000 game shows. Your program should select a door at random in which the car is placed. Simulate the user choosing a random door and simulate revealing the door in which there is no prize. The user should be allowed to choose whether to keep the first selection (made by random by the computer) or choose the other door (only ask the user one time per simulation). Sum up the result in the end. What is the best thing to do? Keep the first door selected or choose the other door?

5

5.1 Project

An approximate value of π can be calculated using the Gregory-Leibniz Series which is

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \ldots + (-1)^n \frac{1}{2n+1}$$

Write a class called "Pi". Add a function to set the order of the series and another function to calculate pi. Also use a member variable to store the result of the calculate function.

5.2 Project

The Maclaurin series of a function approximates the function around $x = 0$. The series of the function e^x is given by

$$e^x \approx 1 + x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \frac{1}{4!}x^4 + \ldots + \frac{1}{n!}x^n$$

where n is the the order.

First, create a function that can calculate the factorial of a number and then create a function that can approximate the value of e^x at a specific point x and with a given order n. The final program should allow the user to calculate the value of e^x at different values of x and n. The user should be allowed to repeat the calculation for different values of x and n.

HINT: Use the pow-function from the cmath-header file (and compare with the exp-function to check your result).

6

6.1 Project

According to the wikipedia article `https://en.wikipedia.org/wiki/Determination_of_the_day_of_the_week` it is possible to calculate the day of the week by the

following formula:

$$w = \left(d + \lfloor 2.6 \cdot m - 0.2 \rfloor + y + \left\lfloor \frac{y}{4} \right\rfloor + \left\lfloor \frac{c}{4} \right\rfloor - 2c\right) \mod 7$$

where Y is the 4-digit year (minus 1 if the month is January or February), y is the last two digits of Y, c is the first two digits of Y, d is the day of the month and m is the shifted month number (March = 1, February = 12). The symbol $\lfloor \rfloor$ denotes the round down operation. The result is given as a number between 0 and 6 where 0 represents sunday and 6 represents saturday. In cases where w is negative, add 7 to the result.

Write a class called Date with the following declaration

```
1  class Date {
2  public:
3      Date();
4      Date(int year, int month, int day);
5
6      void setYear(int year);
7      void setMonth(int month);
8      void setDay(int day);
9
10     int getDayOfWeek();
11 private:
12     int _year, _month, _day;
13 };
```

where the method `getDayOfWeek()` calculates the day of the week according to the above formula.

6.2 Project

Write a program that simulates the roll of 5 dice and the sum must be calculated for each roll. A number of rolls n to be simulated is selected by the user.

7

7.1 Project

Extend project 6.2 with a for-loop and an array such that you can calculate a histogram of the sum of the dices (values 5 to 30). Print the histogram.

7.2 Exercise

Write a class called ArrayInt100 which contains an array as a member variable. The array should be an integer array with a capacity of 100 elements. The class should have functions to

- fill the array with random numbers (between the parameters a and b)
- find the maximum number in the array.
- print the contents of the array.
- to count the number of occurrences of a specific value given as a parameter to the function (e.g. 2's).
- to swap the contents of the array. The function should swap the first element with the last element, the second element with the second last element and so forth. All elements in the array are swapped. Verify that the function works for both even number of elements in the array and uneven number of elements in the array.

7.3 Exercise

Create a small program that stores a set of phone numbers. Write a class for a Person holding at least the name and phone number of one person. Create a class called PhoneBook and use a member variable (a vector) to store the persons in the phone book. Write a function that can find a specific person given the name of that particular person. Consider the search criteria to be incomplete (e.g. allow a part of the name to be entered).

Allow the user to search for persons in the phone book.

8

9

9.1 Låneberegning

Opgaven her går ud på at skrive et program, der kan udføre renteberegninger på annuitetslån.

Krav til opgaven

Alle klasser, der implementeres, skal have en tilhørende header-fil. Klasserne skal som minimum overholde de anførte deklarationer.

Klassen Loan

I skal udvikle en klasse "Loan" (husk stort L), som kan beregne omkostningerne på et annuitetslån. I Listing 1 er klassedefinitionen givet.

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

        // Skal initialisere Loan til at have
        Loan(double debt, int years, int paymentsPerYear, double
            interestRate);

        // Returns the number of years the loan lasts
        int getYears() const;
        void setYears(int years);

        // Amount of payments per year (terminer)
        int getPaymentsPerYear() const;
        void setPaymentsPerYear(int paymentsPerYear);

        // Debt
        double getDebt() const;
        void setDebt(double debt);

        // Rente
        double getInterestRate() const;
        void setInterestRate(double rate);

        // Calculate the total interest of a loan for all the
            years
        double totalInterest() const;

        // Calculate the total repayment of a loan including the
            interests
        double totalPayment() const;

        // Calculate the total net interest of a loan after tax
            refund
        double totalInterestTaxDeducted(double taxDeductionRate)
            const;

        // Output the periodical payments with unpaid balance,
            paid interest and repayment of each payment to stream
            object ost
        void outputPeriodicalPayments(std::ostream& ost) const;
private:
        double mDebt, mInterestRate;
        int mYears, mPaymentsPerYear;
}
```

Listing 1: Loan interface

Beregning af ydelse

Et annuitetslån kendetegnes ved, at man betaler en fast ydelse for hver termin (altså hver gang man betaler rente og afdrag på lånet). Typisk tilbagebetales et kreditforeningslån over 30 år eller mindre og betalingen foregår fire gange årligt.

Renten er fastsat af kreditforeningen og oplyses i "p.a." - altså pro anno (per år). Renten tilskrives fire gange årligt samtidig med man betaler ydelsen. Ydelsen består altså af både afdrag og renter (herunder skal også indregnes bidrag som en procentsats).

For at beregne ydelsen på lånet kan man bruge flg. formel

$$y = G \cdot \frac{r}{1 - (1 + r)^{-n}}$$

hvor G kaldes for hovedstolen (gælden), r er renten per ydelse og n er det totale antal terminer. Dvs. ved 4 terminer per år og et lån på 3% i rente, der løber over 30 år bliver $n = 4 * 30 = 120$ og $r = 3/4\% = 0.75\% = 0.0075$ (husk heltalsdivision i C++ kan give andre resultater).

Hovedprogram

Lav et hovedprogram, som lader brugeren indtaste de nødvendige oplysninger og fortæl brugeren hvad vedkommende skal betale i renter over lånets løbetid og hvad den totale tilbagebetaling bliver.

Fradrag

Renten man betaler på sit lån er fradragsberettiget som et ligningsmæssigt fradrag. Det betyder, at man ca. får 30.6% af renteudgiften tilbage i skat (afhængig af kommunen man bor i). Lav en ekstra funktion, som beregner de totale renter med denne skattefordel. Funktionen kaldes `double`totalInterestTaxDeducted(double taxDeductionRate) const .

Ydelsesforløb

Lav en tabel over ydelsesforløbet, hvor man har flere kolonner med restgæld samt renteudgift og tilbagebetaling for hver ydelse.

Der skal være mulighed for at streame til andet en blot std::cout, så funktionen skal kaldes som `void outputPeriodicalPayments (std::ostream&` ost) const. Output skrives til `ost`-objektet. Tallene bør skrives med to betydende cifre efter kommaet. Anvend "bankers rounding" til afrunding.

Flere låntyper

Indskriv rente- og bidragssatser for forskellige låntyper. Lav nu beregningerne på de forskellige låntyper og lad så brugeren se beregningerne for alle de forskellige låntyper. Se evt. `https://www.totalkredit.dk/boliglan/kurser-og-priser/` for prisblad på omkostninger, bidragssatser på forskellige låntyper samt de gældende renter.

Anvendelse af pointere

Mindst en hjælpefunktion skal implementeres på en sådan måde, at der konstrueres et array som angives som parameter til en funktion - eksempelvis til beregning af ydelsesforløbet. Dertil benyttes altså pointere og indeksering kan foretages vha. pointere og ikke indeks i array. Opgaven dækker en stor del af pensum (indtil nu).

10

10.1 A vector with n elements

Finish the implementation of the n-dimensional mathematical vector with the header file:

```
1  class Vector {
2  public:
3      Vector(int n = 0);
4      Vector(const Vector& v);
5
6      virtual ~Vector();
7
8      Vector& operator=(const Vector& rhs);
9      bool operator==(const Vector& rhs);
10     bool operator!=(const Vector& rhs);
11
12     Vector operator+(const Vector& rhs);
13     Vector operator-(const Vector& rhs);
14
15     Vector& operator+=(const Vector& rhs);
16     Vector& operator-=(const Vector& rhs);
17
18     double& operator[](int i);
19     const double& operator[](int i) const;
20
21     friend std::istream& operator>>(std::istream& input, Vector& a
           );
22     friend std::ostream& operator<<(std::ostream& output, const
           Vector& a);
23
24     friend double dot(const Vector& l, const Vector& r);  // Dot
           product
25 private:
26     int mSize;
27     double *mElems;
28 };
```

Write a program that uses the vector and tests all functions.

What happens if you don't implement copy constructor and copy assignment operator. Try to test the following code and see what happens?

```
    Vector a(3);
    {
        Vector b(2);
        b[0] = 1;
        b[1] = 2;
        Vector a;
        a = b;
    }
    Vector c(3);
    c[0] = 0;
    c[1] = 0;
    c[2] = 0;
    std::cout << a << std::endl;
    std::cout << "Finish" << std::endl;
```

11

11.1 Shapes 1

Create a class called `Shape` with a protected member variable of type double called `mArea` and a member function called `area()` that returns the value. Now create three classes `Circle`, `Rectangle` and `Triangle` which inherits from the `Shape` class. The three classes should calculate the area and save it to the member variable `area`.

Write a program that creates different shapes and calculates the area of these.

12

12.1 Shapes 2

Create a class called `Shape` a member function called `area()`. Now create three classes `Circle`, `Rectangle` and `Triangle` which inherits from the `Shape` class.

The Shape class should now have the declaration as given in Listing 2

```
class Shape {
public:
    // Default constructor
    Shape();

    // Constructor with name
    Shape(const std::string& name);

    // Name
    void setName(const std::string& name);
    const std::string& getName() const;

    // Area
```

```
14      virtual double area() = 0;
15
16  private:
17      std::string _name;
18  };
```

Listing 2: Class Shape

Implement the area function for the Circle, Rectangle and Triangle. Remember to create suitable constructors for these classes and call the proper base-class constructor.

12.2 Ticket Company

A ticket company sells tickets to the Opera. In the opera you can have different type of seats. The basic ticket and the floor ticket in front of the stage. The floor tickets have a higher price than the standard ticket.

Write a class called `BasicTicket` which has the following member variables: `seat`,ticketno, city, opera and `price`. Create set/get functions for these variables. Write a class called `FloorTicket` that inherits from the `BasicTicket` class. Override the function `getPrice()` and use the BasicTicket's `getPrice()` function and return the value multiplied by 1.3. It is important that you use the the BasicTicket's `getPrice()` function.

Write a program that tests the basic ticket and floor tickets.

13

14

14.1 Expense management

The tool "pdftotext.exe -layout netto.pdf" (Xpdf tools from xpdfreader.com) converts the pdf to a text file. This specific file contains a digital receipt from the Danish Supermarket Netto. Construct a program that reads this input from a text-file. The program should categorize the expenses into a set of categories like: food, beverages, sweets, cleaning, electronics, tools, etc.

Consider creating a second file with the "trigger word" followed by the category. E.g.

```
laks food
havregryn food
salt food
```

```
matador sweets
cola beverages
toiletpapir cleaning
ajax cleaning
computer electronics
```

Construct the program such that you can automatically detect the receipt sender (e.g. Bilka, Netto etc) and then according to the layout of their receipts register the expenses.

If no category exist for an expense item, a warning should be stated on the screen.

Obtional: The program should be able to read all pdf-files in a directory, call the pdftotext software and convert to txt-files. Read all textfiles and show the total expenses in different categories.

14.1.1 Example netto receipt

```
        MØLLEVEJ 2
     5260 ODENSE S.

HAVFRISK LAK         99,95

RABAT                30,95-

FINVALSET HAVREGRY 11,95

ØGO HVEDEMEL 2 KG 13,95

       TOTAL 94,90

DANKORT              94,90

MOMS UDGØR           18,98

       Du blev betjent af:
                    julie

 5 1 139 05 03 19 16:50
Butik 7566 MOMSNR.35954716

     KIG FORBI WWW.NETTO.DK
       OG WWW.JOB.NETTO.DK

         8-22 ALLE UGENS 7 DAGE
```

Netto er en del af Salling Group
- som er 100% dansk og ejet af

Salling Fondene. Og fordi vi er
fondsejet går en del af vores

overskud til gode formål i samfundet.

Salling Group er det nye navn for
Dansk Supermarked

2019-03-05 16:50

Køb DKK 94,90

Dankort

Contactless

XXXX XXXX XXXX 1111

Term: 70970803-583178

NETS A/S

5848881

KC1 Nets no:0005848881

ATC:00119 AED:000000

AID: A0000001214711

PSAM: 5374978-0000544228

ARC:00 STATUS:0000

Aut. kode: 165026

REF:583178 Autoriseret

Kortholders kopi

```
NETTO
MØLLEVEJ 2
5260 ODENSE S.
```

14.2 Training session

A 34 year old male person with a weight of 90 kg, a height of 185cm, and a VO2max of 48 participates in an indoor bike session. In this session, the participant measures his heartrate approximately every second. After the session, the person would like to calculate the approximate net amount of calories burned. The formula for calculating the gross calories burned is given by[1]

$$cal_m = ((-95.7735 + (0.634H) + (0.404V) + (0.394W) + (0.271A))/4.184) \cdot t/60$$
$$cal_f = ((-59.3954 + (0.45H) + (0.380V) + (0.103W) + (0.274A))/4.184) \cdot t/60$$

where H is the current heart rate, V is the VO2max, W is the weight, A is the age, and t is the time spent with that particular heart rate measured in seconds.

Write a class to store the heart rate and the amount of time spent in that heart rate sample measured in seconds.

```
1  class HeartRate {
2  public:
3      HeartRate();
4      HeartRate(int hr, int time);
5
6      void setHeartRate(int hr);
7      void setTime(int time);
8
9      int getHeartRate();
10     int getTime();
11 private:
12     int _hr, _time;      // Heart Rate and time in seconds.
13 };
```

Write one more class to store a training session (use a `std::vector` for the heart rate data and the time).

```
1  class TrainingSession {
2  public:
3      TrainingSession();
4
5      void readData(std::string filename);
6
7      double calcCalorieBurnNet();
8      double calcCalorieBurnGross();
9      double calcBMR();
10
```

[1] http://www.shapesense.com/fitness-exercise/calculators/heart-rate-based-calorie-burn-calculator.shtml

```
11      double totalTime();
12
13  private:
14      std::vector<HeartRate> _hr;
15  };
```

You might need more functions and member variables to set the persons age, weight, and VO2max.

The method calcCalorieBurnNet() should calculate the net calorie burn of the complete training session, calcCalorieBurnGross() calculates the gross calorie burn and the method calcBMR calculates the BMR of the individual person (see formula in project 3.3).

The net calorie burn can be calculated using

$$G - \frac{\text{BMR} \cdot T}{24}$$

where G is the gross calorie burn, and T is the total amount of time of the training session in hours.

In this project, the input file looks like:

```
Time,HR (bpm),Cadence,Temperatures (C)
00:00:00,136,101,28.4
00:00:01,136,101,28.4
00:00:02,136,103,28.4
00:00:03,136,103,28.4
00:00:04,136,102,28.4
00:00:05,137,100,28.5
00:00:06,137,101,28.4
00:00:07,138,99,28.5
00:00:09,139,99,28.5
00:00:10,140,100,28.4
00:00:13,140,97,28.5
00:00:14,139,96,28.5
00:00:15,139,96,28.4
00:00:16,140,98,28.5
```

The first line consist of headlines of each data element. The second line starts the data samples. You might have noticed, that the sample from seconds 8, 11, and 12 are missing in the input data. It is OK to assume, that the Heart Rate is constant if samples are missing in the input data.

A complete training session file will be provided from your instructur. You should answer the question: what is the total net calorie burn for that training session. Write the result as a comment in the hand-in.

14.3 Accounting

I denne opgave skal man skrive et program, som kan identificere udgifterne som findes i ens bankkonto. Udgifterne skal grupperes efter udgiftstype. Summen for hver udgiftstype beregnes.

Datafiler

Datafilerne er typisk (fra min netbank) i et format med flg.

```
DATO DATO TEKST BELØB SALDO
```

Ved at åbne dokumentet i excel og eksportere det som en tekstfil separeret med tabulatortegn fås følgende:

```
31-01-2017 31-01-2017 kontaktløs Dankort Bilka Odense - 391,25  3.763,04
31-01-2017 31-01-2017 kontaktløs Dankort Harald Nyborg A/S - 108,90  4.154,29
31-01-2017 31-01-2017 WWW.ALIEXPRESS.COM, LONDON USD 3,72 - 26,35  4.263,19
31-01-2017 31-01-2017 kontaktløs Dankort SuperB Bellinge - 20,00  4.289,54
31-01-2017 31-01-2017 kontaktløs Dankort Fakta 91 - 9,50  4.309,54
30-01-2017 30-01-2017 kontaktløs Dankort SuperB Bellinge - 91,00  4.319,04
30-01-2017 30-01-2017 kontaktløs Dankort Eurest kantinen - 8,00  4.410,04
27-01-2017 27-01-2017 Salon Centrum - 265,00  4.418,04
27-01-2017 27-01-2017 Dankort F24 - 8246 - 262,70  4.683,04
26-01-2017 26-01-2017 Dankort-køb pakkelabels.dk/ - 46,25  4.945,74
24-01-2017 24-01-2017 PAYPAL *FONE STUFF, 35314369001 - 49,20  4.991,99
23-01-2017 23-01-2017 Visa/Dankort Modpost EUR TOMMYHILFIGER4  607,52  5.041,19
23-01-2017 23-01-2017 kontaktløs Dankort SuperB Bellinge - 91,90  4.433,67
23-01-2017 23-01-2017 Dankort Elgiganten 3030  47,00  4.525,57
20-01-2017 20-01-2017 kontaktløs Dankort Carls Jr. Odense - 158,00  4.478,57
20-01-2017 20-01-2017 kontaktløs Dankort Carls Jr. Odense - 30,00  4.636,57
20-01-2017 20-01-2017 kontaktløs Dankort Elgiganten 3030 - 47,00  4.666,57
20-01-2017 20-01-2017 kontaktløs Dankort Eurest kantinen - 8,00  4.713,57
19-01-2017 19-01-2017 Dankort-køb reservedelsshop.ha   105,00  4.721,57
18-01-2017 18-01-2017 Fru Lund - 20,00  4.616,57
16-01-2017 16-01-2017 kontaktløs Dankort SuperB Bellinge - 41,95  4.636,57
16-01-2017 16-01-2017 Dankort-køb incover.dk - 89,25  4.678,52
16-01-2017 16-01-2017 kontaktløs Dankort Fakta 91 - 47,45  4.767,77
16-01-2017 16-01-2017 Dankort-køb Boozt.com  500,85  4.815,22
13-01-2017 13-01-2017 kontaktløs Dankort Rema 1000 Skt. Kle - 39,40  4.314,37
13-01-2017 13-01-2017 Adidas, Amsterdam - 598,00  4.353,77
12-01-2017 12-01-2017 Dankort-køb wattoo.dk - 1.179,19  4.951,77
12-01-2017 12-01-2017 Dankort-køb ComputerSalg A/S - 319,73  6.130,96
12-01-2017 12-01-2017 Visa/Dankort Modpost EUR TOMMYHILFIGER4 201,94  6.450,69
11-01-2017 11-01-2017 SuperB Bellinge - 47,00  6.248,75
11-01-2017 11-01-2017 Ovf fra løn/budget  3.000,00  6.295,75
09-01-2017 09-01-2017 kontaktløs Dankort Rema 1000 Skt. Kle - 54,90  3.295,75
09-01-2017 09-01-2017 kontaktløs Dankort SuperB Bellinge - 43,95  3.350,65
06-01-2017 06-01-2017 TOMMYHILFIGER - 809,66  3.394,60
```

```
04-01-2017 04-01-2017 Dankort-køb freetrailer.dk - 39,70  4.204,26
04-01-2017 04-01-2017 kontaktløs Dankort Eurest kantinen - 8,00  4.243,96
03-01-2017 03-01-2017 Dankort-køb Boozt.com - 500,85  4.251,96
03-01-2017 03-01-2017 kontaktløs Dankort Eurest kantinen - 30,00  4.752,81
02-01-2017 02-01-2017 kontaktløs Dankort SuperB Bellinge - 100,00  4.782,81
02-01-2017 02-01-2017 kontaktløs Dankort Fakta 91 - 14,95  4.882,81
02-01-2017 02-01-2017 Dankort 1-2-3 BELLINGEVEJ,- 321,51  4.897,76
02-01-2017 02-01-2017 kontaktløs Dankort SuperB Bellinge - 100,95  5.219,27
```

Program

main-metoden i programmet kunne se sådan ud:

```
1  int main() {
2      const string filename = "f:/bank-januar.txt";
3      ifstream ifs(filename);
4      if (ifs.fail()) {
5          cout << "Could not open file" << endl;
6          exit(1);
7      }
8      vector<string> lines = readLines(ifs);
9      ifs.close();
10
11     ifs.open("f:/ownCloud/map.txt");
12     if (ifs.fail()) {
13         cout << "Could not open file" << endl;
14         exit(1);
15     }
16     vector<string> mapLines = readLines(ifs);
17     ifs.close();
18
19     vector<pair<string, string> > map = loadMap(mapLines);
20     vector<vector<string> > table = splitLines(lines);
21
22     // Separate data from the map
23     vector<string> dates = getDates(table);
24     vector<string> texts = getTexts(table);
25     vector<double> amount = getAmounts(table);
26
27     // Set the types according to the map
28     vector<string> types = getTypes(texts, map);
29
30     std::ofstream ofs("f:/output.txt");
31     if (ofs.fail()) {
32         std::cout << "Could not open file for writing" << std::
               endl;
33     }
34     ofs.setf(std::ios::fixed);
35     ofs.precision(2);
36     for (unsigned int i = 0; i < table.size(); ++i) {
37         ofs << dates[i] << "\t" << types[i] << "\t" << texts[i] <<
                "\t" << amount[i] << endl;
38     }
39     ofs.close();
```

```
40 |
41 |     return 0;
42 | }
```

Listing 3: int main()

Funktionen readLines

Funktionen skal importere data fra parameteren ifs angivet i funktionen

```
1 | std::vector<std::string> readLines(std::ifstream& ifs)
```

Output er en vektor med strings.

Funktionen splitLines

Anvendes til at opdele hver linje fra kontoudtog til en vektor af elementer af strings.

```
1 | std::vector<std::vector<std::string> > splitLines(std::vector<std
  |     ::string>& lines)
```

Funktionen returnerer en vektor af en vektor som indeholder strings (altså en tabel). Dvs. output bliver nu:

```
DATO DATO TEKST BELØB SALDO
DATO DATO TEKST BELØB SALDO
DATO DATO TEKST BELØB SALDO
...
...
```

Input til funktionen er bare output fra funktionen readLines.

Funktionen loadMap

Ideen med denne funktion er, at den anvender et sæt af søgeord som havner i en kategori. F.eks. kan ordet "Fakta" resultatere i kategorien "Dagligvarer". Denne funktionen tager et input som er en vektor hvori der ligger en streng på formen:

```
ORD KATEGORI
```

hvor ORD og KATEGORI er adskilt med tabulator-tegn.

Et eksempel på en sådan fil dette er indlæst fra kunne være:

```
Eurest Kantine
Carls Jr. Café og restaurant
Fakta Dagligvarer
SuperB Bellinge Dagligvarer
Rema 1000 Dagligvarer
Fakta Dagligvarer
Fru Lund Dagligvarer
Bilka Odense Dagligvarer
Adidas Tøj og Sko
TOMMYHILFIGER Tøj og Sko
Boozt.com Tøj og Sko
incover.dk Småanskaffelser
1-2-3 Brændstof
F24 Brændstof
Shell Brændstof
Salon Centrum Frisør
Elgiganten Småanskaffelser
ComputerSalg Småanskaffelser
*FONE STUFF Småanskaffelser
WWW.ALIEXPRESS.COM Småanskaffelser
```

Selve funktionen returnerer en vektor af "pair" med to strings. Se evt. dokumentation for pair her: `http://www.cplusplus.com/reference/utility/pair`

```
1 std::vector<std::pair<std::string, std::string> > loadMap(std::
      vector<std::string>& lines)
```

Funktionen getDates, getTexts, getAmounts

De tre funktioner har næsten samme virkemåde. De henter de relevante data ud af parameteren table og returnerer en vector med data:

```
1 std::vector<std::string> getDates(std::vector<std::vector<std::
      string> >& table)
2 std::vector<std::string> getTexts(std::vector<std::vector<std::
      string> >& table)
3 std::vector<double> getAmounts(std::vector<std::vector<std::string
      > >& table)
```

Funktionen getAmounts har en lidt anden virkemåde, da denne også skal konvertere tekst til tal. Bemærk at mellemrum mellem "-" og tallet skal fjernes. Det samme gælder tusindtalsseparator. Derudover skal den kunne håndtere dansk komma.

Funktionen getTypes

Funktionen anvender det "map" vi har opbygget og kategoriserer udgifterne herefter:

```
std::vector<std::string> getTypes(const std::vector<std::string>&
    texts, const std::vector<std::pair<std::string, std::string> >&
     map)
```

Det er blot tekst-strengen som anvendes fra kontoudtoget til at identificere udgiften. Dernæst anvendes det nævnte map for at kategorisere udgiften. Strategien i metoden er flg:

- For hver tekstlinje søges efter de forskellige nøgleord.
- Hvis ordet findes i teksten, så vælges denne kategori
- Hvis ordet ikke findes i teksten, så sættes en tom streng ind i vektoren (eks. v.push_back(“”))

Output fra metoden er en vektor af samme størrelse (size) som parameteren texts.

Udvidelse til opgaven

Beregn den totale udgift indenfor hver kategori, der eksisterer. Skriv output af denne beregning til skærm og en fil i et let læsbart resultat.

15

15.1 Dictionary lookup

Create a dictionary from the files da.wl, en.wl and de.wl (danish, english, german) using a set. If the value is found in the set, then the word is a valid word. Allow the user to enter a string, e.g. derhterharlished, and search for all possible words in this list. E.g. this string contains (at least) der, har, is, i and er in the danish language. Test all possible substrings by checking the set.

The word list files have one word per line (pure text files).

16

16.1 Merge

Define a function with declaration *mergeArray* that basically does exactly the same thing as the standard algorithm merge.

```
template<class ForwardIt1>
void mergeArray( ForwardIt1 first, ForwardIt1 mid, ForwardIt1 last
    );
```

Listing 4: mergeArray

However, in this case, the merged result, must be stored at the same location as *first* initially points to. You are not allowed to define the function using a for-loop. This means you are only allowed to use standard algorithms from the algorithms header.

Define a second version of the function that as its fourth argument accepts a binary predicate function.

```
template<class ForwardIt1, class BinaryPredicate>
void mergeArray( ForwardIt1 first, ForwardIt1 mid, ForwardIt1 last
    , BinaryPredicate p );
```

Listing 5: mergeArray

To declare a temporary storage of the same type as the iterator points to, the following code lines can be used:

```
    typedef typename std::iterator_traits<ForwardIt>::value_type T
        ;
    std::vector<T> temp(std::distance(first, last));
```

You are not allowed to use the *inplace_merge* function to solve this exercise.

The following test-code can be used:

```
    std::ostream_iterator<double> outputD{std::cout, " "};
    std::vector<double> a{1,2,4,5,1,3,4,7};
    std::cout << "Before merge: ";
    std::copy(a.begin(), a.end(), outputD);
    std::cout << std::endl;

    mergeArray(a.begin(), a.begin()+4, a.end());

    std::cout << "After merge : ";
    std::copy(a.begin(), a.end(), outputD);
    std::cout << std::endl;



    std::vector<double> b{7,4,3,1,5,4,2,1};
    std::cout << "Before merge: ";
    std::copy(b.begin(), b.end(), outputD);
    std::cout << std::endl;

    std::greater<double> g;
    mergeArray(b.begin(), b.begin()+4, b.end(), g);

    std::cout << "After merge : ";
    std::copy(b.begin(), b.end(), outputD);
    std::cout << std::endl;
```

Listing 6: Test code

17

18

18.1 template ArrayList

Opgaven afleveres som tællende aktivitet og udføres i grupper af 2 personer. Det er ikke tilladt at samarbejde med samme person(er) som tidligere tællende aktiviteter i dette semester. Hvis dette finder sted, vil opgaven tælle som 0 point.

I denne opgave skal vi implementere en absktrakt datatype ved navn ArrayList. Det skal implementeres som en template klasse. Koden gemmes i en fil med navn "ArrayList.h".

Vi vil tage udgangspunkt i den mest grundlæggende funktionalitet som kendes fra Java i klassen ArrayList. Se yderligere dokumentation her: `http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html`

18.1.1 Krav til opgaven

Når opgaven er løst skal interface være overholdt og kunne køre main-filen, som vises her:

```
1  #include <iostream>
2  #include "ArrayList.h"
3
4
5  int main() {
6      ArrayList<double> array;
7
8      for (int i = 0; i < 33; ++i) {
9          array.add(i*1.1);
10     }
11
12     array.add(4, -5);
13     array.remove(1);
14
15     for (int i = 0; i < array.size(); ++i) {
16         std::cout << array[i] << ", ";
17     }
18     std::cout << "\b\b  " << std::endl << std::endl;
19
20     std::cout << "ArrayList have reserved space for " << array.
           reserved() << " elements and stores " << array.size() << "
           elements." << std::endl;
21
22     array.trimToSize();
23
24     std::cout << "ArrayList is now trimmed and have reserved space
            for " << array.reserved() << " elements and stores " <<
           array.size() << " elements." << std::endl << std::endl;
```

```
25
26
27      ArrayList<double> subArray = array.subArrayList(1,10);
28
29      std::cout << "Sub ArrayList have reserved space for " <<
            subArray.reserved() << " elements and stores " << subArray.
            size() << " elements." << std::endl;
30
31      for (int i = 0; i < subArray.size(); ++i) {
32          std::cout << subArray[i] << ", ";
33      }
34      std::cout << "\b\b  " << std::endl << std::endl;
35
36      double* sArray = subArray.toArray();
37
38      for (int i = 0; i < 7; ++i) {
39          subArray.add(i+1);
40      }
41
42      // The basic array prints after adding elements to subArray
43      std::cout << "Basic array of sub array contains: " << std::
            endl;
44      for (int i = 0; i < subArray.size()-7; ++i) {
45          std::cout << sArray[i] << ", ";
46      }
47      std::cout << "\b\b  " << std::endl << std::endl;
48
49      // The sub array now has elements
50      std::cout << "Sub array with 7 elements added: " << std::endl;
51      for (int i = 0; i < subArray.size(); ++i) {
52          std::cout << subArray[i] << ", ";
53      }
54      std::cout << "\b\b  " << std::endl << std::endl;
55
56      // The original array has
57      std::cout << "Original array" << std::endl;
58      for (int i = 0; i < array.size(); ++i) {
59          std::cout << array[i] << ", ";
60      }
61      std::cout << "\b\b  " << std::endl;
62
63      // Copy constructor
64      // Copy assignment
65      ArrayList<double> array2 = array;
66      ArrayList<double> subArray2;
67      subArray2 = subArray;
68
69      std::cout << "array2 = ";
70      for (int i = 0; i < array2.size(); ++i) {
71          std::cout << array2[i] << ", ";
72      }
73      std::cout << "\b\b  " << std::endl << std::endl;
74
75      std::cout << "subArray2 = ";
76      for (int i = 0; i < subArray2.size(); ++i) {
77          std::cout << subArray2[i] << ", ";
```

```
78      }
79      std::cout << "\b\b  " << std::endl << std::endl;
80
81
82      // Move assignment operator
83      ArrayList<double> subArray3 = std::move(subArray2);
84      array2 = std::move(subArray2);
85
86      std::cout << "subArray2 = ";
87      for (int i = 0; i < subArray2.size(); ++i) {
88          std::cout << subArray2[i] << ", ";
89      }
90      std::cout << "\b\b  " << std::endl << std::endl;
91
92      std::cout << "subArray3 = ";
93      for (int i = 0; i < subArray3.size(); ++i) {
94          std::cout << subArray3[i] << ", ";
95      }
96      std::cout << "\b\b  " << std::endl << std::endl;
97
98      return 0;
99  }
```

Listing 7: main.cpp

18.1.2 Klassen ArrayList

I skal udvikle en klasse "ArrayList" (husk forskel på store og små bogstaver), som indeholder et array. Til forskel fra et almindeligt array skal klassen håndtere dynamisk allokering af hukommelse. Dette betyder, at når der indsættes flere elementer i "listen", så skal arrayet "udvides". Dette gøres normalt ved at konstruere et nyt array som har dobbelt længde af det gamle og så kopiere elementerne ind i det nye array. Herefter nedlægges det gamle array. Denne funktionalitet skal implementeres i klasse ArrayList som vist herunder.

Klassen skal implementeres som template og overholde flg. interface. Den ene metode kaldet add er implementeret i nedenstående header fil.

Husk at template klasser altid implementeres direkte i headeren.

```
1   #ifndef ARRAYLIST_H
2   #define ARRAYLIST_H
3
4   template <typename T>
5   class ArrayList {
6   public:
7           ArrayList();
8
9           // Copy constructor
10          ArrayList(const ArrayList<T>& c);
11
12          // Move constructor
```

```
13          ArrayList(ArrayList<T>&& c);
14
15          /*
16           * Constructor with a reserved number of elements
17           */
18          ArrayList(int reserved);
19
20          virtual ~ArrayList();
21
22          /*
23           * Copy assignment operator
24           */
25          ArrayList<T>& operator=(const ArrayList<T>& a);
26
27          /*
28           * Move assignment operator
29           */
30          ArrayList<T>& operator=(ArrayList<T>&& a);
31
32
33          /*
34           * Add element to dynamic array
35           */
36          void add(const T& element) {
37                  if (mSize == mReserved)
38                          extendStorage();
39
40                  mElems[mSize] = element;
41                  ++mSize;
42          }
43
44          /*
45           * Inserts the element at placement "idx" in array and
                moves the remaining
46           * items by one place, restoring the old element at "idx".
47           *      check whether it is needed to extend the storage.
48           *      move all elements from _size to idx (reverse) one
                element to the right in the array
49           *      set _elems[idx] equal to the element to be inserted
50           */
51          void add(int idx, const T& element);
52
53          /*
54           * Get a const reference to the element at idx
55           */
56          const T& operator[](int idx) const;
57
58          /*
59           * Get a reference to the element at idx
60           */
61          T& operator[](int idx);
62
63          /*
64           * Removes the element at placement "idx" by moving all
                the remaining elements
65           * by one place to the left in the array
```

```
66            */
67           void remove(int idx);
68
69           /*
70            * Returns the number of elements stored
71            */
72           int size() const;
73
74           /*
75            * Returns the number of items currently reserved in
                 memory
76            */
77           int reserved() const;
78
79           /*
80            * Returns true if number of elements in array is zero
81            */
82           bool isEmpty() const;
83
84           /*
85            * Trims the storage array to the exact number of elements
86            * stored.
87            */
88           void trimToSize();
89
90           /*
91                * Sorts the array using insertion sort (or another
                     algorithm)
92                * You are not allowed to use standard algorithms from
                     algorithm header.
93            */
94           void sort();
95
96           /*
97            * Returns a new ArrayList with elements from "fromIdx"
                 index to "toIdx"
98            */
99           ArrayList<T> subArrayList(int fromIdx, int toIdx) const;
100
101          /*
102           * Returns a new C style array (copy created with new)
                 with all elements
103           */
104          T* toArray();
105
106  private:
107          /*
108           * extendStorage():
109           *       create new array with size 2*_reserved
110           *       copy old data to the new array
111           *       delete old array
112           *       update pointer _elems to point to the new array
113           * (Since this method is private, the method will only be
                 used internally,
114           * but the functionality is needed).
115           */
```

```
116         void extendStorage();
117
118         /*
119          * Member variables
120          */
121         int mReserved;  // The current capacity of "_elems" array
122         int mSize;     // The number of elements stored
123
124         T* mElems;       // Array for storing the elements
125 };
126
127
128 #endif // ARRAYLIST_H
```

Listing 8: ArrayList interface