# Aktiekurser

På fondsbørsen handles der aktier konstant. De handles på forskellige tidspunkter, forskellige mængder og til forskellige priser. Priserne går op og ned i løbet af dagen afhængig af den salgspris og købspris man tilbyder køb og salg til.

Hver enkelt handel registreres og for hvert minut summeres alle handler for det pågældende minut. Her angives normalt forskellige parameter. Den første parameter er ”open”, som angiver kursen på første handel indenfor det pågældende minut, time, hver anden time, dag, uge eller hvor lang en periode man opgør for ad gangen. Herudover angives ”low” som er periodens mindste pris, ”high” som er periodens højeste pris samt ”close” som er sidste handel indenfor perioden, som data er opgjort for.

Et eksempel på minut-opgjorte data er givet herunder for aktien Coloplast:

Local time, Open, High, Low, Close, Volume

03.01.2022 09:11:00.000 GMT+0100,1155.454,1155.454,1155.454,1155.454,0.0014

03.01.2022 09:12:00.000 GMT+0100,1155.428,1155.428,1155.428,1155.428,0.0014

03.01.2022 09:13:00.000 GMT+0100,1155.463,1155.463,1153.458,1153.458,0.0067

03.01.2022 09:14:00.000 GMT+0100,1154.425,1154.942,1154.425,1154.942,0.0027

Med udgangspunkt i handlerne foregået fra 9:14:00 til 9:15:00 (sidste linje) kan vi se at første handel gennemførtes til kurs 1154.425, højeste pris var 1154.942, mindste pris var 1154.425 samt sidste handelspris i minutten var 1154.942. Vi kan altså konkludere at i netop dette minut har prisen været opadgående. Samtidig kan vi se at der er handlet 0.0027 millioner værdipapirer i dette minut.

I denne opgave vil vi kigge kursdata i kommaseparerede filer. Opgaven går ud på at indlæse og behandle filerne i et program.

For at løse denne opgave stilles nogle header-filer til rådighed.

## Klassen DateTime og headeren

Klassen benyttes til at repræsentere en dato og et tidspunkt. Klassen bliver ikke så avanceret, men klassen skal kunne omsætte teksten med dato og klokkeslæt til et tidspunkt. Alle getters og setters skal altså implementeres sammen med funktionen parse.

#ifndef DATETIME\_H

#define DATETIME\_H

#include <string>

class **DateTime**

{

public:

**DateTime**();

**DateTime**(unsigned int year, unsigned int month, unsigned int day, unsigned int hour, unsigned int minute, unsigned int second, int timezone)

: mYear(year), mMonth(month), mDay(day), mHour(hour), mMinute(minute), mSecond(second), mTimezone(timezone) {

}

**DateTime**(const std::string& dateTime) { parse(dateTime); }

//unsigned int getDayOfWeek() const;

void **parse**(const std::string& dateTime);

unsigned int **getYear**() const;

void **setYear**(unsigned int year);

unsigned int **getMonth**() const;

void **setMonth**(unsigned int month);

unsigned int **getDay**() const;

void **setDay**(unsigned int day);

unsigned int **getHour**() const;

void **setHour**(unsigned int hour);

unsigned int **getMinute**() const;

void **setMinute**(unsigned int minute);

unsigned int **getSecond**() const;

void **setSecond**(unsigned int second);

int **getTimezone**() const;

void **setTimezone**(int value);

private:

unsigned int mYear = 0, mMonth = 0, mDay = 0, mHour = 0, mMinute = 0, mSecond = 0;

int mTimezone = 0;

};

#endif // DATETIME\_H

## Klassen TradeInfo og headeren

Klassen TradeInfo skal blot indeholder hver enkelt tidsinterval som læses fra filen og de tilhørende data: åbningskurs, lukkekurs, maksimum og minimum for intervallet samt hvilken volumen der er handlet. Getters og setters skal implementeres her

#ifndef TRADEINFO\_H

#define TRADEINFO\_H

#include "DateTime.h"

class **TradeInfo**

{

public:

**TradeInfo**();

**TradeInfo**(const DateTime& dateTime, double open, double high, double low, double close, double volume)

: mDateTime(dateTime), mOpen(open), mHigh(high), mLow(low), mClose(close), mVolume(volume)

{

}

double **getOpen**() const;

void **setOpen**(double open);

double **getHigh**() const;

void **setHigh**(double high);

double **getLow**() const;

void **setLow**(double low);

double **getClose**() const;

void **setClose**(double close);

double **getVolume**() const;

void **setVolume**(double volume);

DateTime **getDateTime**() const;

void **setDateTime**(const DateTime &dateTime);

private:

DateTime mDateTime;

double mOpen = 0, mHigh = 0, mLow = 0, mClose = 0, mVolume = 0;

};

#endif // TRADEINFO\_H

## Klassen TradeLoader

Klassen TradeLoader skal stå for at indlæse og konvertere alle data til TradeInfo-objekter.

Funktionen load skal som parameter have et filnavn. Funktionen skal så læse filen og konvertere hver linje til et array med 6 strenge og gemme data i mData. Hvis load kaldes flere gange med forskellige filer skal der blot tilføjes til mData member variablen.

Funktionen getTradeInfo skal konvertere data i membervariablen til TradeInfo objekter. Dvs. datoen skal konverteres til et DateTime-objekt, som lagres i et TradeInfo-objekt og alle tal skal konverteres til double-variable, som også gemmes i TradeInfo-objekterne.

#ifndef TRADELOADER\_H

#define TRADELOADER\_H

#include <array>

#include <string>

#include <vector>

#include "TradeInfo.h"

class **TradeLoader**

{

public:

**TradeLoader**();

void **load**(const std::string& filename);

std::vector<std::array<std::string, 6> > **getData**() { return mData; }

std::vector<TradeInfo> **getTradeInfo**();

private:

std::array<std::string, 6> **split**(const std::string& line, char delim);

private:

std::vector<std::array<std::string, 6> > mData;

};

#endif // TRADELOADER\_H

## Klassen Stock og headeren

Klassen Stock bruges til at repræsentere værdien af en aktie over tid. Dvs. man sætter altså alle TradeInfo-objekter, som man har læst med TradeLoader, ind i et TradeInfo-objekt. Det er så meningen, at man skal kunne få fat i TradeInfo-objekter på baggrund af en anden periode end den som er lagret. Dvs. hvis der er læst minut-intervaller fra filen (som i dette tilfælde), så skal disse kunne laves om til time-intervaller eller f.eks. dags-intervaller vha. funktionen getTrades. Husk at getTrades altså skal lave et TradeInfo objekt hvor åbningskurs, lukkekurs samt minimum, maksimum og volumen sættes for perioden.

Det kan her antages at alle data ligger i sorteret rækkefølge, hvis rækkefølgen fra filen overholdes.

#ifndef STOCK\_H

#define STOCK\_H

#include <string>

#include <vector>

#include "DateTime.h"

#include "TradeInfo.h"

class **Stock**

{

public:

**Stock**();

**Stock**(const std::vector<TradeInfo>& tradeInfo) : mTradeInfo(tradeInfo) {}

std::vector<TradeInfo> **getTrades**(unsigned int interval = TRADE\_INTERVAL\_DAY);

std::vector<TradeInfo> **getAllTradeInfo**() const;

void **setAllTradeInfo**(const std::vector<TradeInfo> &tradeInfo);

static const unsigned int TRADE\_INTERVAL\_SECOND = 1; // Do not implement

static const unsigned int TRADE\_INTERVAL\_MINUTE = 2;

static const unsigned int TRADE\_INTERVAL\_HOUR = 3;

static const unsigned int TRADE\_INTERVAL\_2HOUR = 4;

static const unsigned int TRADE\_INTERVAL\_4HOUR = 5;

static const unsigned int TRADE\_INTERVAL\_DAY = 6;

private:

std::vector<TradeInfo> mTradeInfo;

};

#endif // STOCK\_H

## Main og test

Programmet skal testes med nedenstående test-kode:

#include <iomanip>

#include <iostream>

#include "DateTime.h"

#include "TradeLoader.h"

#include "Stock.h"

void **printTrade**(const TradeInfo& ti) {

char oldFill = std::cout.fill();

std::cout.fill('0');

std::cout << std::setw(2) << ti.getDateTime().getYear()

<< "-" << std::setw(2) << ti.getDateTime().getMonth()

<< "-" << std::setw(2) << ti.getDateTime().getDay()

<< " " << std::setw(2) << ti.getDateTime().getHour()

<< ":" << std::setw(2) << ti.getDateTime().getMinute()

<< " ";

std::cout.fill(oldFill);

std::cout << std::setw(10) << ti.getOpen()

<< std::setw(10) << ti.getHigh()

<< std::setw(10) << ti.getLow()

<< std::setw(10) << ti.getClose()

<< std::setw(10) << ti.getVolume() << "\n";

}

void **printTrades**(const std::vector<TradeInfo>& ti) {

for (unsigned int i = 0; i < 24; ++i) {

printTrade(ti[i]);

}

printTrade(ti.back());

}

int **main**() {

TradeLoader sl;

sl.load("../StockTrading/COLOB.DKDKK\_Candlestick\_1\_M\_BID\_01.01.2017-31.12.2017.csv");

sl.load("../StockTrading/COLOB.DKDKK\_Candlestick\_1\_M\_BID\_01.01.2018-31.12.2018.csv");

sl.load("../StockTrading/COLOB.DKDKK\_Candlestick\_1\_M\_BID\_01.01.2019-31.12.2019.csv");

sl.load("../StockTrading/COLOB.DKDKK\_Candlestick\_1\_M\_BID\_01.01.2020-31.12.2020.csv");

sl.load("../StockTrading/COLOB.DKDKK\_Candlestick\_1\_M\_BID\_01.01.2021-31.12.2021.csv");

sl.load("../StockTrading/COLOB.DKDKK\_Candlestick\_1\_M\_BID\_01.01.2022-31.12.2022.csv");

sl.load("../StockTrading/COLOB.DKDKK\_Candlestick\_1\_M\_BID\_01.01.2023-21.03.2022.csv");

std::vector<TradeInfo> tinfo = sl.getTradeInfo();

Stock coloplast;

coloplast.setAllTradeInfo(tinfo);

std::vector<TradeInfo> openHour = coloplast.getTrades(Stock::TRADE\_INTERVAL\_HOUR);

std::vector<TradeInfo> open2Hour = coloplast.getTrades(Stock::TRADE\_INTERVAL\_2HOUR);

std::vector<TradeInfo> open4Hour = coloplast.getTrades(Stock::TRADE\_INTERVAL\_4HOUR);

std::vector<TradeInfo> openDay = coloplast.getTrades(Stock::TRADE\_INTERVAL\_DAY);

std::cout << "-----------------------" << std::endl;

std::cout << "-- Open (hour) --" << std::endl;

std::cout << "-----------------------" << std::endl;

printTrades(openHour);

std::cout << std::endl;

std::cout << "-----------------------" << std::endl;

std::cout << "-- Open (2 hours) --" << std::endl;

std::cout << "-----------------------" << std::endl;

printTrades(open2Hour);

std::cout << std::endl;

std::cout << "-----------------------" << std::endl;

std::cout << "-- Open (4 hours) --" << std::endl;

std::cout << "-----------------------" << std::endl;

printTrades(open4Hour);

std::cout << std::endl;

std::cout << "-----------------------" << std::endl;

std::cout << "-- Open (day) --" << std::endl;

std::cout << "-----------------------" << std::endl;

printTrades(openDay);

std::cout << std::endl;

return 0;

}

## Udvidelse

Implementer en af følgende udvidelser til programmet

* En funktion som beregner det glidende gennemsnit (running average filter) på data som funktionen getTrades på Stock leverer.
  + Det skal være muligt at sætte en parameter som angiver filteres længde
  + <https://en.wikipedia.org/wiki/Moving_average>
* Implementer en anden teknisk indikator
  + Ideer, ZLEMA, Exponential moving average, relative strength index eller lign.
* Lav en visualisering af data
  + Brug evt. kald af eksternt grafprogram som gnuplot, gplot eller lignende
* Lav en grafisk brugergrænseflade til programmet

Prøv evt. også at downloade flere kurser fra <https://www.dukascopy.com/swiss/english/marketwatch/historical/>