WorkdayCalendar class

Løsningen benytter kun C++ Standard Library

struct Time

C++ Standard Library har ikke date/time objekter med operatorer som trengs til containere (set, map, vector). Derfor ble det utviklet en «helper» struct kalt Time. Denne vil bli brukt til WorkdayCalender class.

struct Time(int day, int month, int year, int hour = 0, int minute = 0, int second = 0)

I denne finner en også flags for uke dager som vil bli benyttet.

enum Time::day {sunday, monday, tuesday, wednesday, thursday, friday, saturday };

struct Time benytter funksjonen localtime <time.h> for å beregne datoer.

Struct SimpleDateFormat

For å formatere datoer og klokkeslett :

Struct SimpleDateFormat(std::string dateFormat);

```
void        SimpleDateFormat::set(const std::string& set);
std::string SimpleDateFormat::format(const Time& time);
```

SimpleDateFormat benytter funksjonen strftime <ctime> for å formatere.

Se list over mulige formater: http://www.cplusplus.com/reference/ctime/strftime/

class WorkdayCalender

Denne har ingen argumenter ved deklarering.

Følgende funksjoner er tilgjengelig:

```
void setHoliday            (Time::day     set);  //Sett uke dager (Lørdag/Søndag default)
void setHoliday            (const Time& set);  //Sette  eksakt dato
void delHoliday            (const Time& set);  //Slette eksakt dato
void setRecurringHoliday(const Time& set);  //Sett dato nr. dd.**.****
```

Alle funksjoner over kan kjøres flere ganger og alle blir lagt i lister (set/vector)

void setWorkdayStartAndStop(const Time& start, const Time& stop)

setWorkdayStartAndStop definerer arbeidsdagen (kun klokke verdier benyttes)

Time getWorkdayIncrement(const Time& start, float workingDays)

getWorkdayIncrement er beregningsfunksjonen som returnerer resultatet som struct Time..

Void clear() tilbakestiller objektet til utgangspunktet.

Example

```
SimpleDateFormat format("%d-%m-%Y %H:%M:%S");


WorkdayCalendar workdayCalendar;

workdayCalendar.setWorkdayStartAndStop( Time(0,0,0,8), Time(0,0,0,16) );  //08:00 – 16:00


workdayCalendar.setHoliday          ( Time::friday          );  //Ovale helger
workdayCalendar.setHoliday          ( Time::monday          );

workdayCalendar.setHoliday          ( Time(21,12,2020)      );  //Fri før jul
workdayCalendar.setHoliday          ( Time(22,12,2020)      );
workdayCalendar.setHoliday          ( Time(23,12,2020)      );

workdayCalendar.setHoliday          ( Time(28,12,2020)      ); //Fri før nyttår
workdayCalendar.setHoliday          ( Time(29,12,2020)      );
workdayCalendar.setHoliday          ( Time(30,12,2020)      );


workdayCalendar.setRecurringHoliday ( Time(1,0,0)           ); //Alle innledende måneder
workdayCalendar.setRecurringHoliday ( Time(2,0,0)           );
workdayCalendar.setRecurringHoliday ( Time(3,0,0)           );
workdayCalendar.setRecurringHoliday ( Time(4,0,0)           );
workdayCalendar.setRecurringHoliday ( Time(5,0,0)           );


Time work = workdayCalendar.getWorkdayIncrement(Time(30, 11, 2020), 50);


std::cout << " Workday = " << format.format(work) << std::endl;
```