```
/**
 *  Skjelett for obligatorisk oppgave nr 2 i PROG1003 - OOP.
 *
 *  Programmet er en kalender der man kan legge inn heldags og
 *  tidsbegrensede aktiviteter pÃ¥ spesifikke dager.
 *
 *  Hovedfunksjonalitet:
 *    - Inneholder klassen 'Aktivitet' og dens to subklasser
 *      'Tidsbegrenset' og 'Heldags'. Objekter av de to siste klassene legges
 *      inn for HVER ENKELT DAG inn i to ulike vectorer inni den selvlagede
 *      containerklassen: 'Dag'
 *    - De tre fÃ¸rste klassene inneholder alle constructorer og funksjoner
 *      for Ã¥ lese og skrive alle objektets data.
 *    - 'Dag' inneholder en del ulike funksjoner for Ã¥ hÃ¥ndtere datastrukturen
 *      inni seg. Det er disse medlemsfunksjonene som kalles i fra funksjonene
 *      som startes/kalles fra 'main' for EN gitt dag.
 *    - Den globale vectoren 'gDagene' inneholder ALLE DE ULIKE DAGENE
 *      med hver sine ulike aktiviteter.
 *
 *  @file       OBLIG2.CPP
 *  @author     Sergei Johansen, BPROG
 */


#include <iostream>               //  cout, cin
#include <string>                 //  string
#include <vector>                 //  vector
#include "LesData2.h"
using namespace std;


/*
 *  Enum 'aktivitetsType' (med hva slags aktivitet dette er).
 */
enum aktivitetsType { Jobb, Fritid, Skole, ikkeAngitt };


/**
 *  Baseklassen 'Aktivitet' (med navn og aktivitetstype).
 */
class Aktivitet {
private:
    string navn;
    aktivitetsType kategori;
public:
    Aktivitet() { navn = ""; kategori = ikkeAngitt; }
    void lesData();
    void skrivData() const;
};


/**
 *  Subklassen 'Tidsbegrenset' (med tidspunkter for start/stopp av aktivitet).
 */
class Tidsbegrenset : public Aktivitet {
private:
    int  startTime, startMin, sluttTime, sluttMin;
    bool klokkeslettOK(const int time, const int minutt) const;
public:
    Tidsbegrenset() { sluttMin = sluttTime = startTime = startMin = 0; };
    void lesData();
    void skrivData() const;
};


/**
```

```
 *  Subklassen 'Heldags' (med nÃ¦rmere beskrivelse av aktiviteten).
 */
class Heldags : public Aktivitet {
private:
    string beskrivelse;
public:
    Heldags() { beskrivelse = ""; };
    void lesData();
    void skrivData() const;
};


/**
 *  Selvlaget container-klasse 'Dag' (med dato og ulike aktiviteter).
 */
class Dag {
private:
    int dagNr, maanedNr, aarNr;
    vector <Tidsbegrenset*> tidsbegrensedeAktiviteter;
    vector <Heldags*> heldagsAktiviteter;

public:
    //    Dag()  {  };
    Dag(const int dag, const int maaned, const int aar) {
        dagNr = dag;  maanedNr = maaned;  aarNr = aar;
    };
    ~Dag();
    bool harDato(const int dag, const int maaned, const int aar) const;
    void nyAktivitet();
    void skrivAktiviteter() const;
    void skrivDato() const;
};


bool dagOK(const int dag, const int maaned, const int aar);
Dag* finnDag(const int dag, const int maaned, const int aar);
void frigiAllokertMemory();
void nyAktivitet();
void skrivDager(const bool inkludertAktiviteter);
void skrivEnDag();
void skrivMeny();


vector <Dag*> gDagene;           ///<  Dager med aktiviteter


/**
 *  Hovedprogrammet:
 */
int main() {
    char kommando;

    skrivMeny();
    kommando = lesChar("\nKommando");

    while (kommando != 'Q') {
        switch (kommando) {
        case 'N': nyAktivitet(); skrivMeny();     break;
        case 'A': skrivDager(true); skrivMeny();  break;
        case 'S': skrivEnDag(); skrivMeny();      break;
        default:  skrivMeny();        break;
        }
        kommando = lesChar("\nKommando");
    }

    frigiAllokertMemory();
```

```cpp
    return 0;
}


// ------------------------------------------------------------------------
//                        DEFINISJON AV KLASSE-FUNKSJONER:
// ------------------------------------------------------------------------

/**
 *  Leser inn ALLE klassens data.
 */
void Aktivitet::lesData() {

    std::cout << "\nTast inn navn: ";
    getline(cin, this->navn);

    int type = lesInt("Velg aktivitetstype : (0 = Jobb, 1 = Fritid, 2 = Skole)", 0, 2);

    switch (type)
    {
    case 0:
        this->kategori = Jobb;
        break;
    case 1:
        this->kategori = Fritid;
        break;
    case 2:
        this->kategori = Skole;
        break;
    }
}


/**
 *  Skriver ut ALLE klassens data.
 */
void Aktivitet::skrivData() const {

    std::cout << "\n\tNavn: " << this->navn << std::endl;

    switch (this->kategori)
    {
    case Jobb:
        std::cout << "\tAktivitetstype: Jobb" << std::endl;
        break;
    case Fritid:
        std::cout << "\tAktivitetstype: Fritid" << std::endl;
        break;
    case Skole:
        std::cout << "\tAktivitetstype: Skole" << std::endl;
        break;
    }

}


/**
 *  Leser inn ALLE klassens data, inkludert morklassens data.
 *
 *  @see   Aktivitet::lesData()
 *  @see   klokkeslettOK(...)
 */
void Tidsbegrenset::lesData() {

    Aktivitet::lesData();
```

```
    std::cout << "Velg varighet: " << std::endl;

    this->startTime = lesInt("Starttime", 0, 23);
    this->startMin = lesInt("Startmin", 0, 59);
    this->klokkeslettOK(this->startTime, this->startMin);

    this->sluttTime = lesInt("Sluttime", 0, 23);
    this->sluttMin = lesInt("Sluttmin", 0, 59);

    while (this->sluttTime < this->startTime)
    {
        this->sluttTime = lesInt("Ugyldig. Velg senere tid", this->startTime, 23);
    }


    if (this->sluttTime == this->startTime)
    {
        while (this->sluttMin <= this->startMin)
        {
            this->sluttMin = lesInt("Ugyldig. Velg senere tid", this->startMin, 60);
        }
    }




    this->klokkeslettOK(this->sluttTime, this->sluttMin);
}


/**
 *  Privat funksjon som finner ut om input er et lovlig klokkeslett.
 *
 *  @param   time    -  Timen som skal evalueres til mellom 0 og 23
 *  @param   minutt  -  Minuttet som skal evalueres til mellom 0 og 59
 *  @return  Om parametrene er et lovlig klokkeslett eller ei
 */
bool Tidsbegrenset::klokkeslettOK(const int time, const int minutt) const {

    if (time >= 0 && time <= 23 && minutt >= 0 && minutt <= 59)
        return true;
    else
        return false;
}


/**
 *  Skriver ut ALLE klassens data, inkludert morklassens data.
 *
 *  @see   Aktivitet::skrivData()
 */
void Tidsbegrenset::skrivData() const {         //  Skriver mor-klassens data.

    Aktivitet::skrivData();
    std::cout << "\tVarighet: ";
    std::string nullStartTime;
    std::string nullStartMin;
    std::string nullSluttTime;
    std::string nullSluttMin;

    if (this->startTime >= 10)
        nullStartTime = "";
    else
        nullStartTime = "0";

    if (this->startMin >= 10)
```

```
        nullStartMin = "";
    else
        nullStartMin = "0";

    if (this->sluttTime >= 10)
        nullSluttTime = "";
    else
        nullSluttTime = "0";

    if (this->sluttMin >= 10)
        nullSluttMin = "";
    else
        nullSluttMin = "0";

    std::cout << nullStartTime << this->startTime << ":" << nullStartMin << this->
startMin
    << " - " << nullSluttTime << this->sluttTime << ":" << nullSluttMin << this->sluttMin
<< std::endl;

}


/**
 *  Leser inn ALLE klassens data, inkludert morklassens data.
 *
 *  @see   Aktivitet::lesData()
 */
void Heldags::lesData() {

    Aktivitet::lesData();
    std::cout << "Beskrivelse: ";
    getline(cin, this->beskrivelse);
}


/**
 *  Skriver ut ALLE klassens data, inkludert morklassens data.
 *
 *  @see   Aktivitet::skrivData()
 */
void Heldags::skrivData() const {

    Aktivitet::skrivData();
    std::cout << "\t" << this->beskrivelse << std::endl;
}


/**
 *  Destructor som sletter HELT begge vectorenes allokerte innhold.
 * vector <Tidsbegrenset*> tidsbegrensedeAktiviteter;
    vector <Heldags*> heldagsAktiviteter;
 */
Dag :: ~Dag() {

    for (auto aktivitet : tidsbegrensedeAktiviteter)
    {
        delete aktivitet;
    }
    for (auto aktivitet : heldagsAktiviteter)
    {
        delete aktivitet;
    }

    tidsbegrensedeAktiviteter.clear();
    heldagsAktiviteter.clear();
}
```

```
/**
 *  Finner ut om selv er en gitt dato eller ei.
 *
 *  @param   dag     -  Dagen som skal sjekkes om er egen dag
 *  @param   maaned  -  MÃ¥neden som skal sjekkes om er egen mÃ¥ned
 *  @param   aar     -  Ã…ret som skal sjekkes om er eget Ã¥r
 *  @return  Om selv er en gitt dato (ut fra parametrene) eller ei
 */
bool Dag::harDato(const int dag, const int maaned, const int aar) const {

    if (this->dagNr == dag && this->maanedNr == maaned && this->aarNr == aar)
        return true;
    else
        return false;
}


/**
 *  Oppretter, leser og legger inn en ny aktivitet pÃ¥ dagen.
 *
 *  @see   Tidsbegrenset::lesData()
 *  @see   Heldags::lesData()
 */
void Dag::nyAktivitet() {

    char aktivitet = lesChar("Hvilken aktivitet vil du opprette? (T, H)");

    while (aktivitet != 'T' && aktivitet != 'H')
    {
        aktivitet = lesChar("Velg T eller H!");
    }

    switch (aktivitet)
    {
    case 'T':
    {
        Tidsbegrenset* temp = new Tidsbegrenset();
        temp->lesData();
        this->tidsbegrensedeAktiviteter.push_back(temp);
    }
        break;
    case 'H':
    {
        Heldags * temp1 = new Heldags();
        temp1->lesData();
        this->heldagsAktiviteter.push_back(temp1);
    }
        break;
    }
}


/**
 *  Skriver ut ALLE aktiviteter pÃ¥ egen dato (og intet annet).
 *
 *  @see   Heldags::skrivData()
 *  @see   Tidsbegrenset::skrivData()
 */
void Dag::skrivAktiviteter() const {

    std::cout << "\n\t--- Heldagsaktiviterer ---\n";
    for (auto heldags : this->heldagsAktiviteter)
    {
        heldags->skrivData();
    }
    std::cout << "\n\t--- Tidsbegrensede aktiviteter ---\n";
```

```
    for (auto tidsbegrenset : this->tidsbegrensedeAktiviteter)
    {
        tidsbegrenset->skrivData();
    }

}


/**
 *  Skriver KUN ut egen dato.
 */
void Dag::skrivDato() const {

    std::cout << this->dagNr << "/" << this->maanedNr << "/" << this->aarNr << std::endl;
}


// ------------------------------------------------------------------------
//                       DEFINISJON AV ANDRE FUNKSJONER:
// ------------------------------------------------------------------------

/**
 *  Returnerer om en dato er lovlig eller ei.
 *
 *  @param   dag     -  Dagen som skal sjekkes
 *  @param   maaned  -  MÃ¥neden som skal sjekkes
 *  @param   aar     -  Ã…ret som skal sjekkes
 *  @return  Om datoen er lovlig/OK eller ei
 */
bool dagOK(const int dag, const int maaned, const int aar) {

    if (dag >= 1 && dag <= 31 && maaned >= 1 && maaned <= 12 && aar >= 1990 && aar <=
2030)
        return true;
    else
        return false;
}


/**
 *  Returnerer om mulig en peker til en 'Dag' med en gitt dato.
 *
 *  @param   dag     -  Dagen som skal bli funnet
 *  @param   maaned  -  MÃ¥neden som skal bli funnet
 *  @param   aar     -  Ã…ret som skal bli funnet
 *  @return  Peker til aktuell Dag (om funnet), ellers 'nullptr'
 *  @see     harDato(...)
 */
Dag* finnDag(const int dag, const int maaned, const int aar) {


    for (auto dagen : gDagene)
    {
        if (dagen->harDato(dag, maaned, aar))
            return dagen;
        else
            return nullptr;
    }

}


/**
 *  Frigir/sletter ALLE dagene og ALLE pekerne i 'gDagene'.
 */
void frigiAllokertMemory() {
```

```
    for (auto dag : gDagene)
    {
        delete dag;
    }

    gDagene.clear();
}


/**
 *  Legger inn en ny aktivitet pÃ¥ en (evt. ny) dag.
 *
 *  @see    skrivDager(...)
 *  @see    dagOK(...)
 *  @see    finnDag(...)
 *  @see    Dag::nyAktivitet()
 */
void nyAktivitet() {

    skrivDager(false);
    int dagNr;
    int maanedNr;
    int aarNr;

    std::cout << "Velg dag nr: ";
    std::cin >> dagNr;
    std::cout << "Velg maaned nr: ";
    std::cin >> maanedNr;
    std::cout << "Velg aar nr: ";
    std::cin >> aarNr;

    while (!dagOK(dagNr, maanedNr, aarNr))
    {
        std::cout << "\nUlovlig dato: Prov igjen\n\n";

        std::cout << "Velg dag nr.";
        std::cin >> dagNr;
        std::cout << "Velg maaned nr.";
        std::cin >> maanedNr;
        std::cout << "Velg aar nr.";
        std::cin >> aarNr;
    }
    if (!finnDag(dagNr, maanedNr, aarNr))
    {
        std::cout << "\nOppretter ny dag: " << std::endl;
        gDagene.push_back(new Dag(dagNr, maanedNr, aarNr));
        gDagene.at(gDagene.size() - 1)->nyAktivitet();
        std::cout << "\n\nNy dag og aktivitet er blitt opprettet!" << std::endl;
    }
    else
    {
        finnDag(dagNr, maanedNr, aarNr)->nyAktivitet();
        std::cout << "\n\nNy aktivitet er blitt opprettet!" << std::endl;
    }

}


/**
 *  Skriver ut ALLE dagene (MED eller UTEN deres aktiviteter).
 *
 *  @param   inkludertAktiviteter - Utskrift av ALLE aktivitetene ogsÃ¥, eller ei
 *  @see     Dag::skrivDato()
 *  @see     Dag::skrivAktiviteter()
 */
void skrivDager(const bool inkludertAktiviteter) {
```

```
    if (!gDagene.size())
        std::cout << "\n--- Ingen dager enno ---" << std::endl;
    else
    {
        if (!inkludertAktiviteter)
        {
            for (auto dag : gDagene)
            {
                dag->skrivDato();
            }

        }
        else
        {
            for (auto dag : gDagene)
            {
                dag->skrivDato();
                dag->skrivAktiviteter();
            }
        }
    }
}


/**
 *  Skriver ut ALLE data om EN gitt dag.
 *
 *  @see   skrivDager(...)
 *  @see   dagOK(...)
 *  @see   finnDag(...)
 *  @see   Dag::skrivAktiviteter()
 */
void skrivEnDag() {

    skrivDager(false);
    int dagNr;
    int maanedNr;
    int aarNr;

    std::cout << "Velg dag nr: ";
    std::cin >> dagNr;
    std::cout << "Velg maaned nr: ";
    std::cin >> maanedNr;
    std::cout << "Velg aar nr: ";
    std::cin >> aarNr;

    while (!dagOK(dagNr, maanedNr, aarNr))
    {
        std::cout << "\nUlovlig dato: Prov igjen\n";

        std::cout << "Velg dag nr: ";
        std::cin >> dagNr;
        std::cout << "Velg maaned nr: ";
        std::cin >> maanedNr;
        std::cout << "Velg aar nr: ";
        std::cin >> aarNr;
    }

    if (!finnDag(dagNr, maanedNr, aarNr))
        std::cout << "\nDet finnes ikke en slik dag paa lista" << std::endl;
    else
    {
        finnDag(dagNr, maanedNr, aarNr)->skrivAktiviteter();
    }
}
```

```
/**
 *  Skriver programmets menyvalg/muligheter pÃ¥ skjermen.
 */
void skrivMeny() {
    cout << "\nDisse kommandoene kan brukes:\n"
        << "\tN - Ny aktivitet\n"
        << "\tA - skriv ut Alle dager med aktiviteter\n"
        << "\tS - Skriv EN gitt dag og (alle) dens aktiviteter\n"
        << "\tQ - Quit / avslutt\n";
}
```