```cpp
/**
 *
 *   Rute management system som kan legge til ruter,
 *   slette ruter, vise rutenes varigheter og deres stoppesteder
 *
 *   @file     oving1.cpp
 *
 *   @author   Sergei Johansen, BPROG
 *
 *
 */

#include <iostream>
#include <string>
#include <vector>
#include "LesData2.h"

using namespace std;


const int ANTSTOPP = 11; ///< Totalt antall ulike busstopp

struct Rute {
    vector <string> stopp;      // Rutens ulike stoppesteder
    int ruteNr,                 // Reelt rutenr, f.eks. 42, 165, 718
        totMin;                 // Totalt antall minutter å kjøre på ruten
};                              // (fra første til siste stoppested).

// Prototyper
void skrivMeny();
void skrivStopp();
void skrivRuter();
void skrivNesteStoppesteder(const int stopp);
void nyRute();
bool ruteLesData(Rute &rute);
void ruteSkrivData(const Rute rute);
void slettRute();
void slett();
void slett(const int nr);


vector <Rute*> gRuter;          ///< Pekere til rutene

const vector <string> gBusstopp =
    {
        "Skysstasjonen", "Fahlstroms plass", "Sykehuset",
        "Gjovik stadion", "Bergslia", "Overby", "Nybrua",
        "NTNU", "Kallerud", "Hunndalen", "Mustad fabrikker"
    };

// Minutter mellom stoppestedene
const vector <vector <int>> gMinutter =
    {
        {0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0}, // Skysstasjon
        {3, 0, 3, 0, 0, 0, 3, 0, 0, 0, 4}, // Fahlstrøms plass
        {0, 3, 0, 1, 0, 0, 0, 0, 0, 0, 0}, // Sykehuset
        {0, 0, 1, 0, 3, 0, 0, 0, 0, 0, 0}, // Gjøvik stadion
        {0, 0, 0, 3, 0, 2, 0, 0, 0, 0, 0}, // Bergslia
        {0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0}, // Øverby
        {0, 3, 0, 0, 0, 0, 0, 2, 0, 0, 2}, // Nybrua
        {0, 0, 0, 0, 0, 0, 2, 0, 0, 4, 0}, // NTNU
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, // Kallerud
        {0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 2}, // Hunndalen
        {0, 4, 0, 0, 0, 0, 2, 0, 0, 2, 0}, // Mustad fabrikker
    };
```

```
/**
 *  Hovedprogrammet
 */
int main()
{

    char kommando;

    do
    {
        skrivMeny();
        kommando = lesChar("Velg fra menyen");
        switch (kommando)
        {
        case 'N':
            nyRute();
            break;
        case 'S':
            slettRute();
            break;
        case 'A':
            skrivRuter();
            break;
        case 'B':
            skrivStopp();
            break;

        case 'Q':
            break;

        default:
            cout << "\nVelg fra menyen!" << endl;
            break;
        }
    } while (kommando != 'Q');

    cout << "Avslutter..." << endl;
    slett();

    return 0;
}

/**
 *  Skriver meny med kommandoer
 */
void skrivMeny() {
    cout << endl
         << "N - Ny rute" << endl
         << "S - Slett rute" << endl
         << "A - Skriv alle ruter" << endl
         << "B - Skriv alle busstopp" << endl
         << "Q - Avslutt" << endl << endl;
}

/**
 *  Skriver navn på alle stoppestedene
 */
void skrivStopp() {

    cout << "\n\tAlle stoppestedene:\n" << endl;

    for (int i = 0; i < gBusstopp.size(); i++) {
        cout << "\t" << i + 1 << ": " << gBusstopp.at(i) << endl;
    }
    cout << endl;
}
```

```
/**
 *   Skriver utt alle data for alle rutene
 *
 *   @see ruteSkrivData()
 */
void skrivRuter() {

    cout << endl;

    if (gRuter.size() == 0) {cout << "\nIngen ruter funnet" << endl;}

    for(int i = 0; i < gRuter.size(); i++) {
        ruteSkrivData(*gRuter.at(i));
    }

}
/**
 *  Skriver all data om en rute
 *
 *  @param      rute - Den aktuelle ruten
 */
void ruteSkrivData(const Rute rute) {

    cout << "\tRute nr. " << rute.ruteNr << endl
         << "\tRutens varighet: " << rute.totMin << " minutter" << endl
         << "\tStoppstedene: ";

    for(int i = 0; i < rute.stopp.size(); i++) {

        cout << rute.stopp.at(i);

        if(i == rute.stopp.size() - 1) {break;}
        else {cout << " --> ";}
    }

    cout << "\n\n";

}

/**
 *  Leser inn all data om en rute
 *
 *  @param      rute - Referanse til den aktuelle ruten
 *
 *  @return     Om innlest data var godtatt
 *
 *  @see skrivStopp()
 *  @see skrivNesteStoppesteder()
 */
bool ruteLesData(Rute &rute) {   // Fix godkjenning

    // Velger rute nummeret til den nye ruten
    rute.ruteNr = lesInt("\nTast inn nummeret til ruten", 1, 500);

    skrivStopp();

    const int startSted = lesInt("\nVelg nummeret til rutens startsted", 1, ANTSTOPP);

    // Legg til startStedet til starten av den aktuelle ruten
    rute.stopp.push_back(gBusstopp.at(startSted - 1));

    // Neste mulige stoppesteder fra startStedet
    skrivNesteStoppesteder(startSted);

    // Begynner opptelling av rutens varigheter
    rute.totMin = 0;
```

```
    int forrigeSted = startSted;


    do
    {
        const int nesteSted = lesInt("\nVelg nummeret til neste stoppested (0 -
avslutte)",
                                     0, ANTSTOPP);

        // Minst 2 stoppesteder i ruten - true, ellers false
        if(nesteSted == 0 && rute.stopp.size() < 2) {
            return false;
        }
        else if (nesteSted == 0) {
            return true;
        }



        // Legger til minutter av turen mellom forrige og neste stoppstedene
        rute.totMin += gMinutter.at(forrigeSted - 1).at(nesteSted - 1);

        rute.stopp.push_back(gBusstopp.at(nesteSted - 1));

        // Oppdaterer forrige sted
        forrigeSted = nesteSted;
        skrivNesteStoppesteder(nesteSted);


    } while (true);



    return true;

}

/**
 *  legger til rute til rute-list.
 *  (Rutenummer, stoppesteder, rutens varighet)
 *
 *  @see ruteLesData()
 *  @see ruteSkrivData()
 */
void nyRute() {

    // Legge til ny tom rute
    Rute *rute = new Rute;

    gRuter.push_back(rute);

    // Innlesing resultat (true eller false?)
    bool innlesingRes = ruteLesData(*gRuter.at(gRuter.size()-1));


    // Sletter den tomme ruten, hvis den er ugyldig
    if (innlesingRes == false ) {

        delete gRuter.at(gRuter.size()-1);
        gRuter.pop_back();
        cout << "Innlesingen gikk galt. Prov igjen." << endl;
    }
    else {

        // Viser suksess beskjed
        cout << "\n\tDen " << gRuter.size() << ". ruten er blitt lagt til: " << endl;
```

```
        ruteSkrivData(*gRuter.at(gRuter.size()-1));
    }


}

/**
 *  Skriver ut neste lovlige stoppesteder
 *  fra det aktualle stoppestedet
 *
 *  @param      stopp - Det aktuelle stoppestedet
 */
void skrivNesteStoppesteder(const int stopp) {

    cout << "\n\tAktuelt stoppested: " << gBusstopp.at(stopp - 1) << endl
         << "\n\tNeste mulige stoppesteder:" << endl;


    cout << "\t0 - FERDIG MED VALG AV STOPPESTEDER" << endl;

    for(int i = 0; i < ANTSTOPP; i++) {

      if(gMinutter.at(stopp-1).at(i) != 0)
        cout << "\t" << i + 1 << " - " << gBusstopp.at(i) << endl;
    }
}

/**
 *  Sletter EN, INGEN eller ALLE ruter fra lista
 *
 *  @see    slett()
 */
void slettRute() {
    if (gRuter.size() == 0) {
        cout << '\n';
        cout << "\tIngen ruter funnet...Tilbake til meny" << endl;
        cout << '\n';
    }
    else {

        cout << "\tHvilken rute vil du slette?" << endl;
        for(int i = 0; i < gRuter.size(); i++) {
            cout << "\t" << i + 1 << ": "<< (gRuter.at(i))->ruteNr << endl;
        }
        int ruteIndex = lesInt("\t0: Ingen\n\t-1: Slett alle\n", -1, gRuter.size());

        if (ruteIndex == -1)
            slett();
        if (ruteIndex == 0)
            cout << "\tAvslutter fjerning av ruter. Tilbake til menyen.";
        if (ruteIndex > 0 && ruteIndex < (gRuter.size()+1))
            slett(ruteIndex-1);
        cout << "\n\tSlettet! Tilbake til meny\n" << endl;
    }

}

/**
 *  Sletter alle ruter
 */
void slett() {

    for(int i = 0; i < gRuter.size(); i++) {
        delete gRuter.at(i);
    }
```

```
    gRuter.clear();
}

/**
 *  Sletter en valgt rute
 *
 *  @param      nr - nummeret til ruten
 */
void slett(const int nr) {

    Rute *temp = gRuter.at(nr);

    gRuter.at(nr) = gRuter.at(gRuter.size()-1);
    gRuter.pop_back();

    delete temp;
}
```