Prosjektoppgave

i PROG1003 - Objekt-orientert programmering våren 2022

Frist: ***Fredag 8.april 2022 kl.11:00***
Arbeidsform: *1-3 personer*
Arbeidsinnsats: ***Mye***

Innledning

Dere skal i denne prosjektoppgaven lage et litt større (og utvidet ift. oblig nr.1) program som holder orden på masse stoppesteder for bane og buss i en litt større by.

I hovedsak skal programmet håndtere følgende operasjoner:

- **Skrive** ***alle*** **stoppesteder / ruter**
- **Legge inn / skrive** ***ett*** **stoppested / *en* rute**
- **Endre (ta vekk/legge til) stoppesteder på *en* rute**
- **Lage rutetabell for *ett* stoppested på *en* gitt rute**
- **Hele datastrukturen leses fra/skrives til filer**

Globale variable, klasser og datastrukturen

Programmet *skal kun* inneholde to globale objekter av klassene `Stoppesteder` og `Ruter`.
Se **filene:** **datastruktur.pdf** og **`mainV22.cpp`**

Programmet *skal* (i hvert fall) inneholde de seks klassene med (minst) datamedlemmene:

1. **`Stoppesteder`**:
   - `vector <Stoppested*>`: *usortert* vector med alle mulige stoppesteder (for både bane og buss) overalt i hele byen. Nye stoppesteder legges alltid til bakerst. Ingen stoppesteder endrer noen gang sin posisjon/indeks i vectoren.

2. **`Stoppested`**:
   - `string`: navn
   - `vector <int>`: indeksen/nummeret for «alle» *nabo*stoppestedene
   - `vector <int>`: antall minutter til aktuelt nabostoppested nr.`i` i vectoren over

3. **`Ruter`**:
   - `map <int, Rute*>`: `map` med *alle unikt nummererte* `Rute`'r

4. **`Rute`**:
   - `list <Stopp*>`: alle stoppestedene på ruten. Hvert `Stopp` er en `struct` med to `int`'er: en for stoppestedets *unike* nummer/indeks i vectoren i objektet i pkt.1 (vi lagrer altså *ikke* her en kopi av stoppestedets navn), samt antall minutter fra *forrige* stoppested på ruten (kopi av *ett* av tallene i den andre vectoren i pkt.2)

5. **`Bane`** (subklasse av `Rute`):
   - 2x `int`: en for det *faste* antall vogner som går på vedkommende banestrekning, og en for den *faste* lengden (i meter) på dette banesettet

6. **`Buss`** (subklasse av `Rute`):
   - 2x `int`: en for det *faste* antall sitteplasser og en for det *faste* antall ståplasser
   - `bool`: er *alle* bussene på denne ruta leddbuss eller ei

(Blir nok en del bruk for `virtual` ifm. subklassenes kode…..)

Menyvalg / funksjoner

Det skal lages et fullverdig program som har følgende muligheter/menyvalg/funksjonalitet:
(**NB:** *Hvilken kode som må ligge inni i ulike klassene er det opp til dere å designe/bestemme*)

1. **S N <navn>** Stoppested Ny
   Det leses *nytt* (duplikater får *ikke* forekomme) stoppesteds navn.
   Nytt stoppested evt. opprettes, og det legges inn i datastrukturen.

2. **S A** skriv Alle Stoppesteder
   Om det finnes noen stoppesteder så skrives nummeret og navnene til *alle* disse ut på skjermen (gjerne flere på samme linje, og i rette kolonner).

3. **S E <navn>** skriv alt om Ett Stoppested
   Skriver *alt* (dvs. dets fulle navn, navnene på nabostoppestedene hittil registrert og tiden til hvert av dem) om ett gitt stoppested. *Stoppestedsnavnet kan angis med vilkårlig av store og små bokstaver, og kan gjerne avsluttes straks det er entydig.*

---

4. **R N <nr>** Rute Ny
   Det leses et *nytt* (duplikater får *ikke* forekomme) rutenummer. Ut fra om det er en bane- eller bussrute, så opprettes aktuelt objekt. *Alle* dets data leses inn, *inkludert* stoppestedene (navnene/numrene og tiden mellom) på ruten. Ble det lest inn *mer enn* ett stoppested, så legges den nye ruten inn i datastrukturen (eller slettes det bare igjen).

   Ifm. lesing av rutens stoppesteder, så:
   - leses slike inntil kun ENTER skrives (dvs. blankt/tomt navn)
   - skal store/små bokstaver telle likt i navnet («æøå» må gjerne byttes med «eoa»)
   - kan navnet sluttes å skrive inn straks det er entydig
   - kan man gjerne kun oppgi *nummeret* (i stedet for entydig navn) på stoppestedet
   - kan ikke samme stoppested forekomme flere ganger på samme ruten
   - leses tiden mellom stoppestedet og det forrige, *om* det da ikke er kjent allerede, for da hentes/brukes den tiden som er lagret fra før. *Er tiden ny mellom stoppestedene, så legges den inn i begges stoppenes datastruktur.*

5. **R A** skriv kort Alle Ruter
   Skriver ut *alle* rutenes *hoved*data på skjermen. Dvs. for *hver* rute, skrives på *en* linje: dens nummer, om er bane- eller bussrute, og navnene på endestasjonene.

6. **R E <nr>** Rute Endre
   Ut fra valgt rutenummer, skrives ruten ut på skjermen. Deretter spørres brukeren om vedkommende vil fjerne/slette eller smette inn stoppesteder. I begge tilfeller spørres det om mellom hvilke to *eksisterende* stoppesteder på ruten som utgjør ytterpunktene for valget. (*Kan hende referansene til de to stoppestedene må byttes om, slik at de videre i kodingen kommer i rett innbyrdes rekkefølge.*)
   Er «smett inn» valgt, så *må* stoppestedene komme *rett* etter hverandre (ellers er jo ikke innsmettingen entydig). Det leses inn nye stoppesteder på samme måte som ved kommandoen «R N». Husk at stoppestedet *rett* etter den siste nyinnsmettede kan hende nå må få ny tid fra det *forrige* stoppestedet (om dette ikke da allerede finnes registrert).
   Er «fjern» valgt, så fjernes/slettes *alle* stoppene *mellom* de to angitt, men *ikke* de to angitte. Kan hende den siste av de to stoppene nå også må få lagt inn ny tid ift. den forrige.

   I begge tilfeller skrives til slutt den nye ruten ut på skjermen.

7. **R B <nr>** Rute Beskrivelse
   Ut fra valgt rutenummer, så skrives *alle* rutens data, inkludert lister over rutens stoppesteder *begge veier*. Det skrives *en* linje pr. stoppested med navn, antall minutter fra *forrige* stoppested og *totalt* antall minutter fra startstedet for vedkommende retning.

8. **R T <navn>** RuteTabell for ett stoppested
   Ut fra valgt rutenummer og retning på ruten (frem/tilbake) så skrives ruten i den aktuelle retningen. Deretter angir brukeren et entydig stoppested på ruten. Det spørres så om *lovlige* avganger (time og minutt) fra startstedet *innenfor ett* døgn. Dvs. det må leses når noe starter, når noe slutter og intervallet på avgangene (fra 6 til 120 minutter) i dette tidsrommet.
   All innlesningen avsluttes med tiden: 0 0.
   Ut fra dette skrives rutetabellen ut for det ene aktuelle stoppstedet. Der minutter samme time skrives på en og samme linje.

   Ekstra: Har dere lyst til å gjøre litt ekstra, så kan dere jo utvide dette siste punktet med:
   a) Skrive rutetabellen for *alle* rutens stoppesteder - *til filer*. Navnet på filene skal følge formatet: <rutenr> - (bane/buss) - <navn på stoppested> - (<startnavn>-<sluttnavn>).dta
   b) Lagre unna de innleste dataene om tidene fra start-/endestedet. Blir det senere spurt om dette, kan disse presenteres, og brukeren får valget om å bruke dem, eller å angi andre tider.
   c) Skille på ruteavganger fra start-/endested ift. ukedager, lørdager og søndager – i begge retninger (totalt seks avgangstabeller må da lagres fra start-/endestoppet).

Rimelige verdier for `const`'er (f.eks. rutenes numre, maks tid mellom to stoppesteder, min/maks antall sitteplasser/ståplasser/vogner og banesettlengde), `int`'er (som leses inn fra brukeren) og `enum`'er, og hvilke funksjoner/tjenester de ulike objektene må tilby hverandre (interface), må dere selv finne og bestemme. De aller fleste feilsituasjoner, f.eks. ulovlige kommandoer, ikke-eksisterende navn/numre, tomme containere, …m.m, og dertil egnede meldinger, er stort sett ikke bemerket ovenfor. Dette må også selvsagt gjøres/kodes.

Forslag til `mainV22.cpp` kan hentes ned fra prosjektets hjemmeside. Det må i tillegg *minst* lages en global funksjon med utskrift av lovlige kommandoer.

Tips: ***Kommandoene R E og R T bør være det siste dere lager/koder, når alt annet virker og er ferdig laget/testet. For koden for begge disse er ekstra utfordrende.***

Data til/fra filer

I programmet er det totalt involvert to ulike filer: **STOPPESTEDER.DTA** og **RUTER.DTA**
Hele datastrukturen inni de to globale objektene ligger her lagret på hver sin fil.
Formatene bestemmer dere helt selv.

Prosjekt / multifil-program

Dere *skal* utvikle hele dette programmet som et prosjekt, der programmet er splittet opp i mange ulike filer. Følgende (minst 10) .h-filer må lages:

- en med *alle* const'er
- en med *alle* enum'er
- en med deklarasjon av *alle* 'globale' funksjonsheadinger
- en *pr.klasse* med deklarasjon av dets innhold (datamedlemmer og funksjonsheadinger)
- LesData3.h (ligger ferdig på EKSEMPLER-katalogen)

Følgende (minst 9) .cpp-filer må lages:

- en som inneholder main og definisjon av de globale variablene
- *minst* en fil som inneholder definisjon (innmaten) av *alle* de 'globale' funksjonene
- en *pr.klasse* med definisjon av klassens funksjoner (deres innmat)
- LesData3.cpp (ligger ferdig på EKSEMPLER-katalogen)

Hjelp: Se og lær av filene EKS27*.* på EKSEMPLER-katalogen.

Annet (klargjørende?)

- Det er *aldri* to likenavnede stopp rett etter hverandre på *både* en bane- og en bussrute. Dette da banen som oftest går raskere (færre minutter mellom) enn en buss mellom stoppestedene. Går f.eks. buss og bane mellom Bøler og Oppsal, så er bussen innom andre stoppesteder, og bruker derfor lengre tid, og Bøler og Oppsal blir dermed heller ikke naboer på bussruten.
- Det er mye som her *ikke* skal lages/kodes, f.eks: kunne slette et stoppested/en rute, endre: et stoppesteds navn, en rutes nummer, en rutes andre data (enn selve stoppestedene) eller den faste tiden mellom to stoppesteder (når dette først er lagt inn).
- **NB:** Programmets kun to globale objekter er definert på filen der `main` ligger.
  Når disse trengs brukt i andre filer, så refereres de til vha. `extern` ..... (jfr. EKS_27*.*).
- Denne oppgaveteksten er nok ikke helt entydig og utfyllende på alle punkter/måter.
  Derfor er det mulig at dere må gjøre deres egne klargjøringer/presiseringer/forutsetninger (se pkt.2e under «Innlevering» på neste side).

NB NB NB:

1. ***Behold norske navn på klassene (og deres da respektive h- og cpp-filer).***
   (Det er da så mye enklere for læringsassistenene (LA) å rette mange prosjekter hver, når filer og klasser har samme navn som angitt i dette dokumentet.)
2. ***La absolutt alt av h-, cpp- og andre filer ligge i en og samme (topp)katalog på GitLab.***
   (Dette gjør også alt rettearbeidet for LAene mye enklere.)
3. ***Lever tidlig en lenke til prosjektet i Blackboard*** (se pkt.1 under «Innlevering» nedenfor).
4. ***Testkjør prosjektet i god tid før fristen ved å clone det hele ned til en helt ny katalog, og kjør det derfra.*** (Blir en simulering av hvordan LAene vil kjøre/oppleve programmet.)
5. ***Meld fra (via mail) til emnelærer om gruppen består av bare Mac- eller Linux-brukere.***

Innlevering

Innen fredag 8.april 2022 kl.11:00 *skal* dere ha:

1. **levert en *lenke/adresse* til prosjektet på GitLab via Blackboard.**
   Dette gjøres ved å kopiere/klippe ut SSH-adressen for cloning i GitLab.
   Den limes inn i Blackboard via «Skriveinnsending»
   (og *ikke* via «Legg ved filer» - som hittil ved oblig-innleveringer).

2. **lastet opp (committed) på GitLab:**
   a. **deres fungerende, endelige og siste versjon av koden i prosjektet**
   b. **lagt inn *minst* de obligatoriske testdataene i alle filene (jfr. pkt.7 på websiden om prosjektet)**
   c. **de tre stk. ferdig utfylte testskjemaene (pdf) for SA, RN og RE**
   d. ***en* beskrivelse (pdf) av DTA-filenes *format* og eksempler på utseende**
   e. **evt. *en* egen fil (pdf) med egne presiseringer/forutsetninger**

Gruppe(sam)arbeid

- Sørg for at alle ytre rammer er lagt til rette for et godt og konstruktivt samarbeide.
  Dette gjøres bl.a. best ved å sette opp klare og konkrete grupperegler.
- *Jobb mye, effektivt og målrettet allerede fra første stund* (dvs. start «langspurten» straks).
- Dere velger selv antallet (1-3 stk) i gruppen. Men, uavhengig av gruppeantallet,
  så forventes det at dere kommer i mål med prosjektet (*alt* kodes/gjøres og virker).
- Et gruppearbeid er et gruppearbeid. Enten får *alle* godkjent, eller så får *alle* det ikke.
- Det er *ingen reinnlevering på prosjektet* («second chance»). Enten så holder det man har kodet/laget, eller så gjør det ikke det.

Løkke tæll!

FrodeH