Mittuniversitetet
MID SWEDEN UNIVERSITY

PROJEKT
Trains

Objektorienterad programmering i C++ (DT060G) VT 2019 v1.2

Övergripande mål:

Du ska i projektet visa att du i C++11

- kan strukturera och lösa ett komplext problem
- tillämpa enklare objektorienterad analys för att identifiera klasser och operationer
- tillämpa principer för objektorienterad programmering i ett större program
- lära sig och implementera nytt koncept självständigt
- använda dynamisk bindning
- använda dynamisk minneshantering
- använda 'smart pointers" i C++11
- använda olika typer av generiska algoritmer
- skapa och använda olika "callable objects"
- använda undantag
- använda olika metoder för konvertering
- undvika att upprepa kod där lämpliga anrop istället kan göras
- dokumentera och presentera ett arbete i en rapport

Bakgrund

Det utlandsbaserade järnvägsbolaget *Ironbend Train and Brain Railway Inc*. har lagt ut ett konsultuppdrag för att lösa ett internt informationstekniskt problem. Från interna källor i företaget (som helst vill vara anonyma) har erfarits att det råder totalt kaos. Ingen vet var något fordon finns. Det lär vara ren tur om något tåg kan avgå mot rätt destination vid rätt tidpunkt, om det överhuvudtaget finns något tåg att tillgå. På sikt vill man konstruera ett system för att enkelt administrera sina lok och vagnar samt för att hålla ordning på alla tåg, vilket/vilka lok som drar ett tåg, vilka vagnar som ingår, i vilken ordning de kommer i tåget o.s.v. Ett akut konkurshot vilar likt ett *Damokles svärd* över företaget. Det första steget på företagets väg bort från avgrunden är att skapa ett verktyg för att kunna simulera sin verksamhet.

Uppgiften – beskrivning och specifikation

Du ska konstruera en prototyp till det simuleringsverktyg som *Ironbend Train and Brain Railway Inc* så hett eftertraktar.

Fas 1 – Fordonshierakin och tågen

Den första fasen i projektet är att skapa en objektorienterad modell för tåg, lok och vagnar. Modellen ska baseras på ett antal klasser och relationer mellan dessa. Implementera och testa därefter klasserna.

Utgå från följande fakta och beskrivningar av fordon i systemet:

1. Dragande fordon: diesellok och el-lok

2. Personvagnar: sittvagnar och sovvagnar

3. Godsvagnar: täckta och öppna

4. Tåg. Varje tåg har ett (eller flera) lok samt ett antal vagnar. I samma tåg kan det finnas flera typer av både lok och vagnar.

5. Alla fordon som rullar på rälsen har ett unikt *id-nummer.*

6. Varje tågförbindelse har ett unikt *tågnummer*. Tågnummer är ett *logiskt* begrepp som avser en viss kommunikationsförbindelse vid en viss tidpunkt på dygnet. I begreppet ligger typ av lok, antal vagnar av olika typer, avgångsstation, destinationsstation, ordinarie avgångstid och ordinarie ankomsttid.
   T.ex. tåget med nummer 859 från South Bend Central kl 16.13, ankomst till Tahoma City 19.43 går varje dag och består alltid av ett el-lok och tre sittvagnar. *Däremot säger inte tågnumret något om exakt vilka lok/vagnar som används, det kan variera från dag till dag.*

7. Varje tåg genomgår flera olika *tillstånd* under sin levnad. Tillstånden är
   - NOT ASSEMBLED : tåget existerar endast som logiskt begrepp (se punkt 6 ovan), inga lok/vagnar har kopplats ihop.
   - INCOMPLETE : ett eller flera fordon fattas på stationen, tåget är delvis hopkopplat men inte komplett. Detta kommer att medföra att tåget blir försenat.
   - ASSEMBLED : tågets alla fordon har kopplats ihop
   - READY : taget är klart för avgång från ursprungsstationen.
   - RUNNING : tåget är på väg från sin avgångsstation till sin destinationsstation.
   - ARRIVED : tåget har kommit fram till slutstationen.
   - FINISHED : tåget har plockats isär och de ingående rälsfordonen finns. parkerade på slutstationen och kan användas i andra tåg.

   För betygen A och B ska alla dessa tillstånd hanteras. För lägre betyg behöver förseningar som konsekvens av tillståndet INCOMPLETE inte hanteras. Mer om detta i avsnitten Fas 2 – Simuleringsapplikationen och Bedömning.

8. Ett tåg ska sättas ihop av tillgängliga lok och vagnar på avgångsstationen. Information som alltid ska finnas tillgänglig för ett tåg är bl.a.
    - tågnummer
    - ursprung och destination med tider
    - eventuell försening
    - tillstånd
    - typ för de ingående fordonen i rätt ordningsföljd
    - om tåget är hopkopplat: id-nummer och övrig information för varje ingående fordon i rätt ordningsföljd
    - medelhastighet i km/h beräknad från avgångstid, ankomsttid och avstånd.

9. Följande attribut för de olika fordonstyperna ingår i modellen:
    - Personvagn - antalet sittplatser, Internet ja/nej
    - Sovvagn - antalet bäddar
    - Öppen godsvagn - lastkapacitet i ton och lastyta i kvadratmeter
    - Täckt godsvagn – lastvolym i kubikmeter.
    - El-lok - max hastighet i km/h och effekt i kW.
    - Diesel-lok - max hastighet i km/h och bränsleförbrukning i liter/timme

Utnyttja virtuella funktioner för att kunna hämta och skriva ut detaljerad information om en godtycklig samling lok och vagnar.

Testa dina klasser noga innan du går vidare till nästa fas.

Fas 2 - Simuleringsapplikationen

Fas 2 innefattar utveckling av en applikation där verksamheten kan simuleras. Här ska du använda klasserna från fas 1.
Prototypens tänkta funktionssätt kan kortfattat beskrivas på följande sätt. Variationer och utvidgningar för olika betyg beskrivs i avsnittet Bedömning.

- Körning av ett antal tåg ska simuleras från en viss tidpunkt och fram till en annan tidpunkt under ett dygn. Dygnet börjar 00:00 och slutar 23:59.
- Styrning av simuleringen ska i det enklaste fallet ske genom att användaren själv stegar fram tiden i 10-minutersintervaller.
- För varje tidsintervall ska den aktuella simuleringstiden skrivas ut på format hh:mm. Alla förändringar i tågens aktuella tillstånd som skett under det senaste tidsintervallet ska skrivas ut på skärmen så att händelseförloppet kan övervakas kontinuerligt. Om inga tillståndsförändringar har skett ska bara den aktuella simuleringstid skrivas ut.
- Alla händelser (tillståndsövergångar) enligt punkten ovan ska även loggas till en fil med namnet Trainsim.log.
- Istället för att stega fram nästa tidsintervall ska användaren alltid ha möjligheten att välja att få aktuell information om

    1. Tidtabellen för alla tåg
    2. Ett valfritt tåg: tågnummer, tillstånd, avgångsstation, destination, ingående lok/vagnar o.s.v.
    3. En valfri station: eventuella tåg som finns där och deras tillstånd samt vilka

fordon som finns tillgängliga i stationens fordonspool.
När information om fordon presenteras ska den alltid innehålla id och typ.
Även ändringar av inställningar för simuleringen ska alltid kunna göras, t.ex. intervallängden. Mer om detta under Betygskrav.

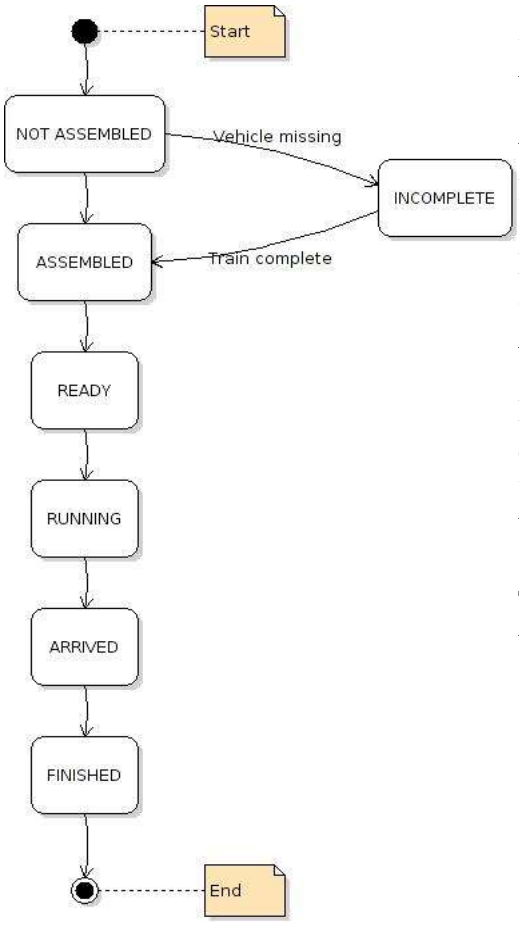
- När simuleringen är slutförd ska statistik presenteras.

På stationerna finns det en pool av olika fordon som ska användas för att bygga upp tåg som ska avgå. Poolens innehåll växlar under simuleringens gång när tåg ska sättas ihop men även när fordonen plockas isär på stationen. Fler tåg med olika tillstånd kan finnas samtidigt på en station.

Ett tåg har en officiell avgångstid som (i bästa fall) även hålls i verkligheten. Under sin livscykel genomgår ett tåg ett antal tillstånd enligt följande:

- Ett tåg börjar alltid i tillståndet NOT ASSEMBLED.

- Det aktuella tågsättet börjar sättas ihop 30 minuter före angiven avgångstid och övergår från tillståndet NOT ASSEMBLED till ASSEMBLED under förutsättning att alla nödvändiga fordon har hittats och tåget har kunnat sättas ihop enligt specifikationen. Vid sammansättning av tåget gäller att om flera fordon av önskad typ finns på avgångsstationen ska fordonet med lägst id väljas.
  Om det fattas ett eller flera fordon får tåget tillståndet INCOMPLETE. Det ofullständiga tåget förblir i detta tillstånd tills fordon som fattas senare eventuellt blir tillgängliga genom att ett ankommande tåg har plockats isär. Tillståndet ändras då från INCOMPLETE till ASSEMBLED.



- Tio minuter före avgångstid övergår tåget från tillståndet ASSEMBLED till tillståndet READY.

- Vid avgångstid inträder tillståndet RUNNING.

- När tåget kommit fram till slutstationen övergår tillståndet från RUNNING till ARRIVED.

- Efter 20 minuter i tillståndet ARRIVED återlämnas de ingående rälsfordonen till fordonspoolen vid slutstationen och tågets tillstånd blir FINISHED.

Tillståndsövergångarna kan beskrivas med tillståndsdiagram till vänster.

Om ett tåg kommer i tillståndet INCOMPLETE innebär det att det blir försenat.
För betygen A och B ska nya försök att göra tåget komplett därefter göras var 10:e minut tills tåget är komplett och kan övergå till tillståndet ASSEMBLED, eller tills simuleringen är klar. För varje misslyckande att göra tåget komplett ska tiderna för avgång och ankomst i senareläggas med 10 minuter. Konsekvenserna av detta är att vissa tåg kommer att anlända försenade till sin destination och att vissa tåg aldrig kommer att lämna sin destination.
För betygen C, D och E behöver inga nya försök att komplettera tåget göras. Konsekvensen blir att ett tåg som hamnat i tillståndet INCOMPLETE aldrig kommer att lämna sin avgångsstation.

Förutsättningarna för simuleringen, dvs. alla data för tåg, stationer, fordon och deras attributvärden ska läsas in vid simuleringens start från de tre datafiler som medföljer projektet. Dessa filer beskrivs under rubriken Datafilerna. Information som ska läsas in är:

- fordonen som finns i systemet, deras unika id , typ och attributvärden.
- stationerna som finns i systemet och vilka fordon som finns på de olika stationerna vi simuleringens start.
- tågen med tågnummer, avgångs- och slutstation , tider, antal och typer av fordon maxhastighet samt avståndet mellan stationerna.

Programmet ska bara känna till namnen på de tre datafiler som ska läsas in vid programstarten, inga övriga simuleringsdata får hårdkodas. Vid inläsningen ska de simuleringsobjekt som motsvarar innehållet i datafilerna skapas dynamiskt.

Simuleringen omfattar ett dygn och startar alltså 00:00. I en ideal värld utan förseningar har alla tåg nått sin destination senast 23:59. Om ett tåg p.g.a. försening har startat men inte kommit fram till sin destination till 23:59 ska simuleringen fortsätta tills tåget har kommit fram och har uppnått tillståndet FINISHED.

Simuleringen avbryts alltså då *något* av följande villkor har uppfyllts:

- alla tåg har uppnått tillståndet FINISHED

- simuleringstiden har passerat 23:59 och tillståndet för alla tåg INTE är RUNNING eller ARRRIVED

Observera att om simuleringen inte är färdig 23:59 får inga tåg sättas samman eller lämna avgångsstationen, vi är ju endast intresserade av tåg som avgått under det angivna dygnet.

Statistik

När simuleringen är slutförd ska statistik presenteras över:
- vilka tåg som kördes och kom fram i tid
- vilka tåg som aldrig lämnade avgångsstationen
För betygen A och B ska även redovisas
- vilka tåg som kördes men var försenade
- förseningen för respektive tåg
- den totala förseningstiden

För betyget A ska dessutom både försening vid avgång och försening vid ankomst anges per tåg och sammanlagt.

Datafilerna

Simuleringsdata läses in från tre textfiler: TrainMap.txt, TrainStations.txt och Trains.txt.

TrainMap.txt
Filen beskriver vilka stationer som finns och avståndet mellan dessa i km. Varje rad innehåller tre fält:
`StationsNamn StationsNamn2 avstånd`
OBS! att alla sträckor är symmetriska och anges därför bara en gång dvs. avståndet a till b är detsamma som avstånder b till a.

TrainStations.txt
Filen beskriver vilka fordon som finns vid de olika stationerna. Varje rad innehåller data för en station med formatet

```
StationsNamn (id typ param0 param1)(id typ param0 param1)...
```

Varje fordon kodas alltså som `(id typ param0 param1)` där `id` är fordonets unika id, `typ` är fordonets typ kodat som ett heltal 0 – 5 med parametrar enligt följande:

Sittvagn = 0;
`param0` = antal sittplatser : int
`param1` = internet ja/nej : int (1 = true / 0 = false)

Sovvagn = 1;
`param0` = antal sängar : int

Öppen godsvagn = 2;
`param0` = lastkapacitet i ton : int
`param1` = lastyta i kvadratmeter : int

Täckt godsvagn = 3;
`param0` = lastvolym i kubikmeter : int

El-lok = 4;
`param0` = max hastighet i km/h : int
`param1` = effekt i kw : int

Diesellok = 5;
`param0` = max hastighet i km/h : int
`param1` = bränsleförbrukning i liter/h : int

Observera att vissa fordonstyper bara har en parameter.

Trains.txt
Filen beskriver tågen. Varje rad innehåller data för ett tåg på formatet

```
trainId avgångsStat ankomstStat avgTid ankTid maxHast typ typ ...
```

`trainId` är tågnumret följt av tågets avgångsstation, ankomststation och maxhastighet. Därefter följer en uppräkning av typer för de fordon som ska ingå i tåget. Kodningen är 0 – 5 enligt ovan. Maxhastigheten anges som ett heltal men bör hanteras som en `double` vid beräkningar.

De tre datafilerna bifogas i två versioner, Windows och UNIX/Linux/Mac.

Tips: Diskret händelsestyrd simulering

Uppgiften lämpar sig väl för ”Discrete Event-Driven Simulation” där alla händelser sker vid diskreta tidpunkter och tiden däremellan kan ”hoppas över”. Ett enkelt ramverk för detta som du kan utnyttja om du vill finns i filen Exempel-DEDS.zip.

Krav på lösningen

- Lösningen ska motsvara beskrivningen och specifikationen enligt ovanstående.
- Lösningen ska efter simuleringen kunna presentera ett korrekt simuleringsresultat. Lösningen ska också tillåta att användaren med hjälp av ett fordons unika id kan följa ett visst fordon från en station via ett tåg till nästa station osv.
- Om din kod innehåller element och ”features” som har funnits i tidigare versioner av C++ men finns i ny version i C++11 så ska C++11-varianterna användas. Inga föråldrade (”deprecated”) språkelement ska användas.
- Koden ska inte innehålla konstruktioner eller element som är specifika för någon viss kompilator eller något speciellt operativsystem, t.ex. `#pragma once` eller `#include<conio.h>`. Villkorlig kompilering för anpassning till olika plattformar är tillåten.
- Minneshanteringen ska garantera att inga minnesläckor uppstår. Allt allokerat minne ska vara återlämnat när programmet avslutas.
- Felkontroller ska alltid göras, returvärden vid funktionsanrop ska alltid kontrolleras och undantag ska fångas. Inga runtime-krascher beroende på felaktig inmatning ska förekomma.
- Koden ska kommenteras med beskrivningar av klasser och funktioner.
- Lösningen ska kompletteras med en projektrapport som beskrivs längre ner.

Bedömning

Projektet ger något av betygen A, B, C, D, E, F eller Fx. Detta betyg blir också det slutliga kursbetyget.

Följande punkter kommer att bedömas

- *Måluppfyllelse*
  Uppfyller programmet beskrivningen av och kraven på lösningen? Ger simuleringen ett korrekt resultat?

- *Struktur*
  Uppdelning i klasser, fördelning av ansvar mellan klasserna.

- *Utnyttjande av språkets möjligheter*
  Användning av genomgångna moment som smarta pekare, generiska algoritmer och ”callable objects”, exceptions, dynamisk bindning mm. Är de valda elementen väl lämpade för uppgiften?

- *Källkoden*
  Konsekvent typografi och indentering beskrivande namn på identifierare samt en kommentering av koden som underlättar förståelsen.

- *Användarvänlighet*
  Informativa utskrifter, enkel inmatning etc.

- *Projektrapport*
  Innehåll enligt specifikation.

I varje avseende som din lösning tydligt avviker från beskrivningen och kravspecifikationen görs ett *påpekande*. Antalet påpekanden påverkar slutbedömningen av projektet.

Betygskrav

En förutsättning för ett godkänt betyg (E) är ett fungerande program enligt beskrivningen i avsnittet Fas 2 – Simuleringsapplikationen. Programmet ska presentera ett korrekt simuleringsresultat utifrån de givna förutsättningarna. Det ska också alltid vara möjligt att kunna få information om vilka fordon som finns i visst tåg eller på en viss station. Man ska alltså kunna följa ett visst fordon under simuleringens gång. Dessutom får lösningen högst ge 5 påpekanden enligt föregående stycke.

Utvidgningar för högre betyg
Specifikationen kan utvidgas för att öka användarvänligheten. Sådana utvidgningar är att under körning erbjuda

a) Möjlighet att välja start- och sluttid inom dygnet för simuleringen.
b) Möjlighet till ändring av intervallängd.
c) Möjlighet att välja mellan att stega fram med fast intervallängd eller att låta simuleringen löpa till nästa händelse.
d) Möjlighet att välja detaljnivå för informationen som lämnas om fordonen, minimum

är id + typ.

e) Möjlighet att för ett visst id får direkt information om var fordonet befinner sig.
f) Möjlighet att för ett viss id få en historik över hur detta fordon har förflyttat sig hittills under simuleringen.
g) Presentation av tågens medelhastighet beräknat från de givna förutsättningarna. Medelhastigheten påverkas dock av förseningar. Tågets hastighet kan ökas i proportion till förseningens storlek för att ordinarie ankomsttid om möjligt ska kunna hållas. Dock kan ett tåg aldrig köras snabbare än sin maxhastighet eller komma fram före sin ordinarie ankomsttid.

För betyget D krävs högst 4 påpekanden samt implementering av a) och b)
För betyget C krävs högst 3 påpekanden samt implementering av a) .. e)

För betygen A och B ska lösningen implementera hantering av försenade avgångar och ankomster genom att regelbundet försöka komplettera tåg som kommit i tillståndet INCOMPLETE enligt tidigare beskrivningar.
Desssutom krävs
För betyget B: högst 2 påpekanden samt implementering av a) .. f)
För betyget A: högst 1 påpekande samt implementering av a) .. g)

Vid inlämning av det färdiga projektet ska du själv ange vilket betyg du siktar på.
Sen inlämning sänker betyget med ett steg.

Rapporten

Skriv en projektrapport som innehåller

- en försättssida med kurs- och projektnamn, ditt namn och id samt datum.
- en kort beskrivning av uppgiften.
- uppgifter om dina utvecklingsmiljö och plattform
- en redogörelse för hur du har löst uppgiften. Fordonshierarkin från fas 1 behöver bara beröras ytligt. Mer intressant är vilka övriga klasser du har skapat och hur du har fördelat ansvar mellan dem och hur du motiverar dina val. Utnyttja gärna UMLs klassdiagram.
- Beskriv gärna ditt testprogram med hjälp av skärm dumpar från testprogram.
- Om du gör projekt i grupp måste du beskriva tydligt att vem gjorde vad i projektet.
- Dina egna kommentarer och slutsatser av arbetet.

Du får även gärna ge kommentarer till kursen som helhet. Jag tar gärna emot konstruktiv kritik och förslag till förbättringar.

Redovisning

Projektet redovisas med programmets källkod, datafiler för simuleringen och en projektrapport i pdf-format. Packa källkod tillsammans med projektrapporten och lämna in den packade filen i projektets inlämningslåda i Moodle.

Genom att du lämnar in detta arbete försäkrar du att det är skapat av dig själv. Du är även ansvarig att se till att det inte finns någon plagierad kod eller text i dokumentet. När du refererar och citerar andra verk måste korrekta källhänvisningar finnas och i fallet citering ska den citerade texten vara tydligt markerad.
http://www.bib.miun.se/student/skriva/referenser

Om plagierad kod eller text finns i dokumentet riskerar du att stängas av från studier. Om samarbete sker utan att detta har stöd i instruktionen för examinationen utgör det normalt en disciplinförseelse och du som student riskerar att stängas av från dina studier.

Lycka till!
Awais