Mittuniversitetet

MID SWEDEN UNIVERSITY

Projektrapport

VT19 DT060G Datateknik GR (B), Objektorienterad programmering i C++, 7,5 hp

Project Trains

Skrivet av: Henrik Henriksson (hehe0601)

MITTUNIVERSITETET

DSV Östersund

Examinator	Awais Ahmad	awais.ahmad@miun.se
Handledare	Peter Bergh	pebe1700@student.miun.se
Författare	Henrik Henriksson	hehe0601@student.miun.se
Utbildningsprogram	Programvaruteknik, 180hp	
Kurs	DT060G Objektorienterad programmering i C++	
Huvudområde	Datateknik	
Termin, år	VT, 2019	

Innehållsförteckning

1 Inledning....4
1.1 Bakgrund....4
1.2 Syfte....4
2 Genomförande....5
2.1 Utrustning....5
2.2 Tillvägagångssätt....5
3 Resultat....6
4 Diskussion....7
Referenser....8

1 Inledning

1.1 Bakgrund

Det internationella tågkompaniet Ironbend Train and Brain Railway inc. Behöver en lösning på deras IT problem där ingen har koll på vilka tågvagnar som finns var, eller om det kommer finnas tillräckliga fordon på en station vid en specifik tid för att ett tåg ska kunna avgå i tid. Tågkompaniet behöver ett verktyg för att kunna simulera och övervaka alla tågavgångar under en 24 timmars period.

Uppgiften är att skapa ett program innehållande en simulator som simulerar alla tågavgångar och tillståndsövergångar ett tåg genomför under en specificerad period. Programmet ska läsa information om tågstationer, tågvagnar och tågtidtabeller från fil. Denna data ska sedan användas i en diskret händelsestyrd simulering som presenteras för användaren och sparas till fil.

1.2 Syfte

Syftet med projektet är att studenten ska kunna uppvisa förmåga att 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. Vidare ska studenten även i projektet uppvisa användning av: dynamisk bindning- och minneshantering, smarta pekare, olika typer av generiska algoritmer, skapa och använda olika ”callable objects”. Studenten ska även kunna uppvisa olika metoder för konvertering och anropsstyrd kod.

2 Genomförande

2.1 Utrustning

För projektets utförande har jag använt mig av Microsoft Visual Studio Community 2017 och Microsoft Visual Studio Community 2019.

2.2 Tillvägagångssätt

Jag påbörjade min lösning med att skissa upp vilka klasser jag skulle behöva. Jag skapade en klass-struktur för tågvagnar där jag utnyttjar en abstrakt klass och virtuellt arv till ett flertal underklasser.

När fordonshierarkin var implementerad identifierade jag vilka andra entiteter som borde finnas, jag landade då vid: stationer, tåg och ett tågsystem. Stationens uppgift är att i sitt stall hushålla för tillfället oanvända tågvagnar. Tåg representerar tåglinjer i en tidtabell med specifikation av vilka tågvagnar de behöver för att kunna slutföra en resa. Tågsystem i det här fallet skulle motsvara ett tågkompani av typ SJ som sammanbinder tåglinjer och tågstationer.

Jag skapade därefter en klass för användargränssnitt, med stödklasser i form av Menyhanterare. När detta var klart började jag strukturera de klasser som behövs för den händelsestyrda simuleringen. Dessa klasser baserades i stort på det lektionsmaterial från ”Burger bar” vi fått till handa inför projektet. För att tillmötesgå syftet har jag nästan exklusivt använt mig av smarta pekare i form av std::shared_ptrs.

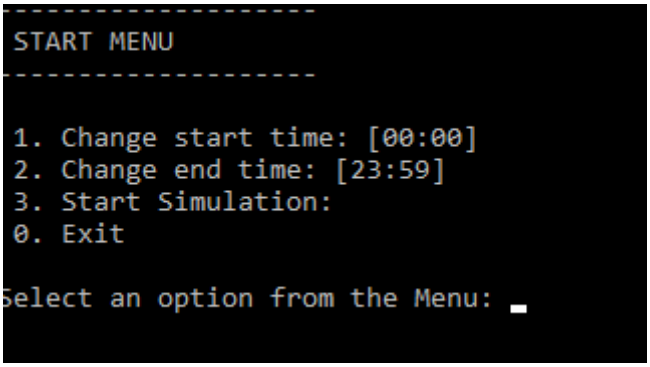
Totalt inkluderas i projektmappen följande filer:
Include: *CarFreightClosed.h, carFreightOpen.h, CarPersonSit.h, CarPersonSleep.h, cfl.h (common functions library), Constants.h, Event.h, LocoDiesel.h, LocoElect.h, Menu.h, MenuItem.h, Simulation.h, Station.h, Train.h, TrainCar.h, TrainMap.h, TrainSystem.h, UserInterface.h.*

Src: *CarFreightClosed.cpp, CarFreightOpen.cpp, CarPersonSit.cpp, CarPersonSleep.cpp, cfl.cpp, Event.cpp, LocoDiesel.cpp, LocoElect.cpp, Main.cpp, Menu.cpp, MenuItem.cpp, Simulation.cpp, Station.cpp, Train.cpp, TrainCar.cpp, TrainMap.cpp, TrainSystem.cpp, UserInterface.cpp.*

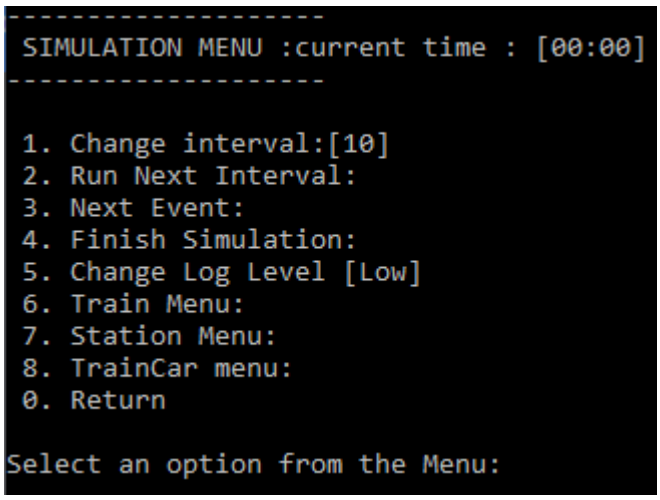
Jag bifogar även filen dt060g_hehe0601v1_projext.exe samt de filer vi fått tillhanda för inläsning: TrainMap.txt, Trains.txt, TrainStation.txt.

3 Resultat

Programmet jag skrivit har funktioner implementerade för att kunna ändra start- och sluttid. Användaren presenteras först en startmeny:



Efter att användaren valt att starta simulationen läses datafilerna in och huvudmenyn presenteras:



I denna menyn kan användaren välja att ändra intervallets längd (som minst 10 minuter), köra hela nästa intervall, köra endast nästa händelse, ändra utskriftsnivå (två nivåer tillgängliga. Jag såg inte ett behov av en ”normal” nivå), få översikt av tåg, stationer och tågvagnar.

Efter en körning av ett intervall kan utskriften se ut så här:

```
Select an option from the Menu: 2
Running next Interval:

00:42 Train: [50]  (ASSEMBLED) from Dunedin 01:12 to
GrandCentral 02:05, Speed: 160km/h, is now Assembled,
 Arriving at the platform at: 01:02

00:44 Train: [17]  (ASSEMBLED) from Liege-Guillemins 01:14 to
GrandCentral 02:37, Speed: 181km/h, is now Assembled,
 Arriving at the platform at: 01:04

00:44 Train: [115]  (ASSEMBLED) from Hauptbahnhof 01:14 to
GrandCentral 02:17, Speed: 196km/h, is now Assembled,
 Arriving at the platform at: 01:04

00:46 Train: [1]  (ASSEMBLED) from GrandCentral 01:16 to
Liege-Guillemins 02:46, Speed: 167km/h, is now Assembled,
 Arriving at the platform at: 01:06

00:46 Train: [34]  (ASSEMBLED) from ST.Pancras 01:16 to
GrandCentral 02:30, Speed: 180km/h, is now Assembled,
 Arriving at the platform at: 01:06

00:47 Train: [99]  (ASSEMBLED) from Shinjuku 01:17 to
GrandCentral 01:57, Speed: 177km/h, is now Assembled,
 Arriving at the platform at: 01:07

Press any key to continue . . .
```

I undermenyn för tågvagnar kan en vagn eftersökas med dess unika ID, eller få information om samtliga tågvagnar i systemet. En enkel utskrift av en tågvagn ser ut likt:

```
Enter a Train Carriage ID to search for: 1
[Coach Carriage] ID: 1, Available seats: 100, Internet Access: Available.
Press any key to continue . . .
```

Medan en mer detaljerad utskrift ger följande:

```
Enter a Train Carriage ID to search for: 1
[Coach Carriage] ID: 1, Available seats: 100, Internet Access: Available.
00:00 Connected to train pool at GrandCentral
04:51 Disconnected from train pool at GrandCentral station
04:51 Connected to train 4
06:06 Disconnected from train 4
06:06 Stabled in train pool at MilanoCentrale station
06:15 Disconnected from train pool at MilanoCentrale station
06:15 Connected to train 71
08:17 Disconnected from train 71
08:17 Stabled in train pool at Luz station
09:34 Disconnected from train pool at Luz station
09:34 Connected to train 89
11:03 Disconnected from train 89
11:03 Stabled in train pool at Hauptbahnhof station
12:48 Disconnected from train pool at Hauptbahnhof station
12:48 Connected to train 124
15:15 Disconnected from train 124
15:15 Stabled in train pool at ST.Pancras station
18:22 Disconnected from train pool at ST.Pancras station
18:22 Connected to train 47
21:01 Disconnected from train 47
21:01 Stabled in train pool at Hauptbahnhof station
Press any key to continue . . .
```

En detaljerad utskrift av ett tåg i ”INCOMPLETE” state kan efter simulationen se ut likt:

```
------------------------------------------------------
 Train: [32]  (INCOMPLETE) from Liege-Guillemins 21:40 to
ST.Pancras 22:52, Speed: 199km/h

Attached Train Carriages:
[Diesel Locomotive] ID: 87, Max Speed: 216 km/h, Fuel Consumation: 636 l/h

Missing Train Carriages:
[Coach Carriage]
------------------------------------------------------
[Coach Carriage]
------------------------------------------------------
[Coach Carriage]
------------------------------------------------------
```

När hela simuleringen körts får användaren automatiskt upp statistikmenyn:

```
--------------------
STATISTICS MENU :
--------------------

1. Change Log Level: [High]
2. Print Statistics:
3. Train Menu:
4. Station Menu:
5. Train Car Menu:
0. Return:
```

Här kan användaren få exempelvis statistiken för en tågvagn som visades ovan. En utskrift av statistiken ger antalet tågvagnar på varje station vid simulationens början samt vilka tåg som aldrig lämnat stationen.

```
------------------------------------------------------------
Number of Train Carriages at the start of the simulation
------------------------------------------------------------
Dunedin = 101
GrandCentral = 102
Hauptbahnhof = 104
Liege-Guillemins = 87
Luz = 77
MilanoCentrale = 108
ST.Pancras = 83
Shinjuku = 79
------------------------------------------------------------
Number of Trains That never left the Station:
------------------------------------------------------------
------------------------------------------------------------
Total: 9
------------------------------------------------------------
```

Sammanfattningsvis har jag implementerat följande uppgifter:

- Möjlighet att kunna välja start- och sluttid för simuleringen.
- Möjlighet att ändra tidsintervall.
- Möjlighet att kunna välja mellan att stega fram mellan fast intervall eller låta simuleringen löpa till nästa händelse.
- Möjlighet att i samtliga menyer välja detaljnivå på informationsutskrift.
- Möjlighet att få direkt information om var ett tåg befinner sig genom dess unika ID (vid station, anslutet till ett tåg).
- Möjlighet att via ID få historik över hur ett unikt fordon förflyttats under simulationen.

4 Diskussion

Tyvärr måste jag med hänsyn till tidsbrist endast eftersträva ett C betyg. Det hade varit roligt att skriva för ett högre betyg, och jag har idéer på hur detta skulle kunna implementeras. Min uppfattning är att det här är ett väldigt omfattande projekt. Det var under utvecklingen ofta svårt att greppa precis allt som skulle behövas för dess utförande. De svårigheter jag stötte på var huruvida underklasser skulle vara tysta. Jag fick hjälp av detta genom lektionsmaterialet för lektion 9, där överlagring av std::ostream objekt exemplifieras. Vidare fann jag det mycket svårt att iterera genom mina fordonsvektorer under Train och Station då jag endast hade vektorer av shared_ptrs att jobba med. Jag fick då hjälp av en tredje års elev som rekommenderade att se över <map> biblioteket. Detta bibliotek med funktionerna std::map och std::pair fann jag oerhört lämpliga för projektets utförande. Jag valde därför att helt skriva om min kod för att utnyttja fördelarna std::map ger då vi i samtliga behållare har ett unikt id (eller namn för stationer) att använda som nyckel, samt sorteringen std::map gör för att försäkra att den tågvagn med lägst id blir valt vid tågbygge.

Det har ibland varit svårt att tolka den information vi fått gällande vilka utskrifter som ska ske under ett specifikt tidsintervall. Den utskrift som sker i min version av simulatorn är min uppfattning av vilka utskrifter som ska vara med. Om användaren sätter ett tidsintervall mellan 12:00-23.59 kommer endast tågövergångar som inträffar under denna period att redovisas, med undantag för tågövergången ”Finished” där jag lagt till 25 extra minuter eftersom det sista tåget som anländer strax innan 24:00 annars aldrig skulle printas.

Slutligen vill jag tillägga att jag önskar att kursen täckt dessa vid något tillfälle då de verkligen underlättade uppbyggnaden av behållare för de olika objekten.

Referenser

[1] Mittuniversitetet, ”Lektion 9-shared_ptr, RTTI, mer om arv och dynamiskt bindning mm.
Tillgänglig: https://elearn20.miun.se/moodle/pluginfile.php/612336/mod_book/chapter/15185/Lektion9.pdf
[Senast åtkomst 2019-05-31]

[2] https://www.geeksforgeeks.org/map-associative-containers-the-c-standard-template-library-stl/