Rapport heislab

I denne rapporten skal vi se på vårt arbeid med å lage styresystemet til en heis. Vi skal se på arkitekturdesign, moduldesign, implementasjon og testing. Til slutt skal vi diskutere hvor god løsningen vår er.

1 Arkitekturdesign

Vi har delt opp heisprogrammet i fire moduler. *Elevfunk*-modulen styrer heisens funksjoner. Den inneholder funksjoner for hvilke lys som skal være på, hvilke knapper som er trykket og lukking/åpning av døra. Modulen *orders* håndterer bestillinger. Den lagrer hvilke knapper som er trykket på og legger til nye bestillinger. Modulen *elevdrive* kontrollerer selve heisen og innholder koden vi trenger for å få heisen til å kjøre, stoppe og være i obstruksjonsmodus. Den holder også styr på hvor heisen er og hvor den sist var. Den siste modulen *globalvar* innholder de globale variablene våre.

Figur **??** viser et klassediagram for systemet. Her ser vi hvordan modulene henger sammen og arver fra hverandre og hvilke funksjoner som er i hver modul.

Figur **??** viser et tilstandsdiagram for systemet. Etter initialisering er heisen i tilstanden *waiting for orders*. Heisen står da i ro og venter på en bestilling. Om en bestilling blir gjort begynner heisen å kjøre og går over i tilstanden *driving*. Heisen kjører til den kommer til riktig etasje og går over i tilstanden *open door*. Etter tre sekunder er timeren ferdig, døra lukkes og heisen går tilbake til *waiting for orders*. *Stop* er tilstanden heisen er i om stoppknappen er trykket inn. Det er mulig å havne i stoppmodus fra alle tilstandene om stoppkappen trykkes på. *Obstruction* er tilstanden heisen er i når man ønsker at heisdøra skal forbli åpen.

Figur **??** viser et sekvensdiagram for systemet. For å gjøre diagrammet mer oversiktlig er ikke lysene tatt med. Vi antar at de ikke er sentrale for forståelsen av hvordan programmet virker, og skrus av og på inne i de andre funksjonene. De grå boksene viser når funksjonen kjører. Den lange grå boksen under orders viser til at *checkIfToBeAdded()* og *getAnyOrders* kjøres hele tiden, men koden går bare videre om de returnerer *True*. I tillegg bes det merkes at mellomrommet mellom *openElevDoor()* og *closeElevDoor()* viser til at døra holdes åpen i tre sekunder.

Arkitekturen vi har valgt gir god oversikt over de forskjellige funksjonene heisen har. Vi har valgt å putte de globale variablene i en egen modul fordi da kan både *orders* og *elevfunk* ha tilgang til de uten at noen moduler trenger å arve begge veiene fra hverandre.

2 Moduldesign

Den første modulen, *globalvar*, inneholder tre globale variabler. Vi har valgt å putte disse i en egen modul for å tydeliggjøre at de er globale og for å holde kontroll de. Disse tar vare på verdier det er viktig at huskes gjennom hele programmet. For at heisen skal kunne huske hvilke bestillinger som er gjort har vi laget matrisen *ordersMatrix*, med to kolonner og ti rekker. Den første kolonnen skal inneholde hvilken etasje en bestilling hører til og den andre hvilken type knapp som ble trykket på. Det er maksimalt mulig med ti ulike bestillinger når heisen har fire etasjer og en matrise med ti rekker er derfor tilstrekkelig for å til en enhver tid kunne lagre alle bestillingene som er gjort. Vi valgte denne løsningen for køsystem fordi det gir en enkel og grei oversikt over hvilke bestillinger som er gjort og det er en enkel kode å jobbe med.

Modulen *orders*, som arver fra både *globalvar* og *elevfunk* inneholder alle funksjonene som endrer på ordersMatrix. Her har vi valgt å ha en funksjon som registrer at en bestilling er gjort (en knapp er trykket på) og iterer gjennom *ordersMatrix* for å sjekke om denne bestillingen allerede ligger lagret. Om bestillingen ikke er lagret går koden videre til *addToOrders()* som legger bestillingen til på den første ledige plassen i matrisen. For å passe på at heisen tar med seg bestillinger som er på veien kjører funksjonen *checkOrdersThisFloor()* hver gang heisen kommer til en etasje. Den iterer gjennom matrisen og sjekker om det det er trykket på en heispanelknapp for denne etasjen eller om noen har trykket på etasjepanelknappen i riktig retning. Om funksjonen finner noen slike bestillinger fjernes den og bestillingene under flyttes oppover så alle de ledigeplassene i matrisen til en hver tid er nederst.

I modulen *elevDrive* er alle funksjonene som får heisen til å kjøre samlet. Her er *driveElevator* hovedfunksjonen som kaller de fleste andre funksjonene og får heisen til å faktisk kjøre dit den skal. Funksjonen tar inn etasjen til den øverste bestillingen i *ordersMatrix* og setter heisens retning til nedover om den befinnes seg over dette eller oppover om den befinner seg under. Denne funksjonen kjøres i en while-løkke i main, så den kjøres på nytt hele tiden. Når heisen kommer til en etasje sørger *driveElevator()* for at først funksjonen *checkOrdersThisFloor()* kjøres og om den finner noen bestillinger for denne etasjen kjøres så *hasReachedTargetFloor()*. Denne funksjonen åpner døra og holder den åpen i tre sekunder med bruk av funksjonen *time*. For å gjøre det mulig for heisen å ta imot bestillinger mens døra er åpen har vi kjørt en nedtelling til 3 sekunder som betingelsen i en while-løkke og puttet funksjonene for å legge til bestillinger og sjekke om stoppknappen og obstruksbryteren er trykket på inni.

3 Testing

For å teste koden startet vi med å teste en og en funksjon. For hver funksjon vi skrev lagde vi en test for å sjekke at akkurat denne funksjonen virket som tenkt. Deretter satte vi de sammen og testet at køsystemet virket for seg og at kjøring av heisen virket for seg.

For å teste køsystemet lagret vi knappetrykk i *ordersMatrix* og printet matrisen for å se at bestillingene ble lagret som de skulle. For å teste at kjøringen av heisen virket som den skulle sendte vi inn forskjellige destinasjoner til kjør-heis funksjonene og så at heisen da kjørte dit vi ville, stoppet og åpnet og lukket døra.

Til slutt satt vi sammen hele systemet og testet at heisen da oppførte seg som den skulle. Denne måten å teste på gjør det letter å fange opp hvor feilene ligger og rette opp om det er noe som ikke virker.

For å teste hele systemet sammen startet vi med å teste at heisen starter i en gyldig tilstand. En gyldig tilstand er definert som at heisen står i en etasje. Under heisens initialisering skal programmet sjekke om heisen er i en etasje eller mellom to. Om heisen er mellom to etasjer, skal den kjøre ned til den er i en. På den måten starter heisen alltid i en gyldig tilstand. Programmet skal nå vite hvilken etasje heisen er i og lagre verdien. Resten av koden skal håndtere bestillinger og kjøring av heisen og skal ikke bli kjørt før tilstanden er gyldig. For å teste at dette virker som det skal startet vi heisen mellom to etasjer. Den kjørte da ned og stopper i en etasje. Krav O1 og O2 er da oppfylt.

Det neste vi testet var heisens evne til å ta imot bestillinger. Vi startet med å teste for en bestilling. For å gjøre dette startet vi heisen i første etasje og prøvde både å trykke på 2 på heispanelet og nedover på etasjepanelet. I begge tilfellene kjørte heisen til andre etasje og stoppet der. Etasjelyset byttet fra 1. etasje til 2. etasje når heisen kom fram og døra åpnet seg i 3 sekunder før den lukket seg igjen. Dette bekreftet at heisen virker som den skal om kun en bestilling gjøres.

Deretter ønsket vi å teste for flere bestillinger. For å gjøre dette trykte vi på flere knapper samtidig og så at alle knappene vi trykte på begynte å lyse. Vi så da at heisen gikk til etasjen vi trykte på først. På veien gikk den forbi etasjer der etasjepanelknappene var trykt på i motsatt retning, men stoppet om de var trykt på i samme retning. Man så også at den stoppet om heispanelknappen var trykt på i etasjen. Når heisen stoppet i en etasje slukkes knappene i den etasjen. Vi så også at etasjelysene lyste opp og slukket som de skulle når heisen kjørte mellom etasjene, og at dørlyset kun lyste når heisen sto i ro i en etasje. For å dobbeltsjekke at alle bestillingene ble lagret og heisen tok de i riktig rekkefølge printet vi *ordersMatrix* underveis og fulgte med på den. Når alle bestillingene

var tatt stoppet heisen med lukket dør. Kravene H1-H4, L1-L5, D1, D2, S1 og S2 er dermed oppfylt.

For å teste at stoppknappen virker som den skal startet vi med å sjekke oppførselen til heisen om vi trykte på den mellom to etasjer. Vi så at heisen stoppet med en gang og lyset på alle knappene slukket. Så lenge vi holdt inne stoppknappen lyste den og ingen andre knapper lyste opp om vi trykte på de. For å dobbeltsjekke at alle bestillingene ble fjernet og ingen nye lagt til printet vi *ordersMatrix* og vi så at den ikke inneholdt noen bestillinger. Når vi slapp knappen fortsatte heisen å stå i ro til en ny knapp ble trykt på og den startet å kjøre dit.

Deretter testet vi å trykke på stoppknappen mens heisen sto i en etasje. Den oppførte seg da likedan som i tilfellet over borsett fra at den åpnet døra og holdt den åpen så lenge knappen ble holdt inne. Når vi slapp den forble døra åpen i tre sekunder før den lukket seg. Fra disse to testene kan ser vi at krav L6, D3 og S4-S7 er oppfylt.

Obstruksjonsbryteren skal bare påvirke programmet når heisdøra er åpen. Den er åpen i to situasjoner, om heisen har håndert en bestilling i en etasje eller om stoppknappen er trykket mens heisen er i en etasje. For å teste dette vippet vi først ned obstruksjons-bryteren mens heisen kjørte og så at ingenting skjedde. Vi vippet så ned bryteren mens heisen sto i en etasje med døra åpen og så at døra ble værende oppe så lenge bryteren var nede. Mens bryteren var nede trykket vi på noen nye knapper for å teste at heisen fortsatt tar imot bestillinger. For å dobbeltsjekke at heisen fortsatt lagrer bestillingene som er gjort printet vi *ordersMatrix* og så at alle bestillingene som var gjort fortsatt var lagret der. Da vi slapp bryteren holdt døra seg åpen i tre sekunder før den lukket seg og heisen begynte å kjøre igjen. Krav D4 og R1 er da oppfylt.

Det er noen flere krav for hvordan heisen skal oppføre seg som vi ikke har nevnt over. Disse kravene er grunnleggende for at heisen skal virke og blir indirekte oppfylt under testene vi har gjennomført. Vi har derfor ikke gjort egne tester på de.

4 Diskusjon

For å implementere køsystemet vårt har vi valgt å bruke en global matrise. Køsystemet er da statisk minneallokert. En ulempe med dette er at matrisen vår alltid har samme størrelse. Den er laget for fire etasjer og vil kun kunne lagre alle mulige bestillinger til en heis med maks fire etasjer. Heisen vil fortsatt virke for flere etasjer, men om det gjøres for mange bestillinger på en gang vil ikke matrisen ha plass til å lagre alle bestillingene som gjøres. Dette vil likevel ikke være et stort problem fordi det er enkelt å utvide matrisen

med å endre på initialiseringen av den. En annen ulempe er at *ordersMatrix* kan ha ti rekker med tomme bestillinger, som betyr at vi bruker unødvendig minne. Dette hadde imidlertid vært et større problem om vi hadde lite minne tilgjengelig. En fordel med å ikke bruke dynamisk minneallokering er at vi ikke risikerer minnelekkasje. I tillegg hadde koden blitt mye mer komplisert om vi hadde brukt f.eks lenkede lister.

I utgangspunktet skal man unngå å bruke globale variabler. Alle funksjoner i programmet har tilgang til å endre på de. Det kan derfor bli et problem å ha kontroll på hvor de blir endret og dermed miste kontroll på innholdet i de. Vi vurderte at i dette tilfellet var globale variabler likevel en grei løsning siden det er et forholdsvis lite og oversiktlig program der det er lett å holde orden på at bare de funksjonene som skal kunne endre på variablene faktisk gjør det. Samtidig gjorde globale variabler det lettere å både lese, forstå og feilsøke koden.

En annen svakhet med programmet vårt er at vi bruker mye for-løkker. Når man iterer gjennom knappene for å se hva som er trykket på kjøres en dobbel for-løkke. Doble for-løkker har lang kjøretid, og fører til at programmet bruker lengre tid enn det kunne ha gjort. Det itereres også ofte gjennom *ordersMatrix* med en for-løkke. Vi mener likevel at det ikke er et stort problem fordi C er et raskt språk. Om programmet skal eskaleres opp så det virker på mange etasjer vil det få større betydning fordi det vil være nødt til å iterere gjennom en mye lengre matrise.

For å passe på at heisen forblir i obstruksjonsmodus om bryteren er nede eller stoppmodus om knappen er trykt inn brukes while-løkker. Det er også brukt while-løkker for å få ønsket oppførsel på dørtimeren. Vi får da while-løkker i while-løkken i *main*. Dette utgjør en svakhet til programmet fordi det kan gjøre det tregere og øker sjansen for at det kan sette seg fast og ikke komme seg ut igjen av en while-løkke. En bedre løsning på dette kunne vært å skrevet koden som en tilstandsmaskin. Da kunne vi brukt *switch* til å bytte mellom de forskjellige tilstandene vist i tilstandsdiagrammet i figur **??**. Dette ville også gjort koden mer leselig og lettere å forstå. Noe som igjen ville gjort det letter å feilsøke og vedlikeholde den.

Selv om programmet har noen svakheter, mener vi fortsatt at vi har en god løsning. Vi har flere erfaringer å ta med til neste heislab. Både hva som funket bra, og hva som kunne blitt gjort bedre.