Prosjektoppgave

i "IMT1082 - Objekt-orientert programmering" våren 2013

Frister:	***Tirsdag 23.april 2013 kl.10.00***
	NB: *Fredag 22.mars kl.09.00 (1.delinnlevering)*
	Tirsdag 9.april kl.09.00 (2.delinnlevering)
Arbeidsform:	*Gruppe (tre (evt. to) personer – fire er uaktuelt)*
Arbeidsinnsats:	***Mye***

Innledning

Dere skal i denne prosjektoppgaven lage et litt større program som holder orden på reservasjoner/bestillinger/bookinger i ulike romkategorier på ulike hoteller.

For å gjøre dette mest mulig reelt (men ikke *for* komplisert), så:

- er til enhver tid dataene om kun *ett* hotell innlest i datamaskinens primærhukommelse.
- inneholder hvert hotell romkategoriene: singel, dobbel og suite.
- inneholder hvert rom en liste med de reservasjonene som er for dette rommet.
- inneholder nåværende rombeboers reservasjon evt. de regningene (Pay-TV, Internet, telefon, (mini)bar, restaurant, …..) han/hun har "pådratt" seg.

I hovedsak skal programmet håndtere følgende operasjoner:

- Lese inn/bytte til et hotell (i primærhukommelsen)
- Foreta en reservasjon eller avbestilling
- Inn- /utsjekking
- Registrere en regning på et rom
- Endre reservasjon
- Oversikt over en del ulike situasjoner/data (se kommandoen 'O' nedenfor)

Globale variable, klasser og datastrukturen

Programmet trenger *kun* å inneholde de fire globale variablene:

- en peker til *ett* `Hotell`-objekt
- *ett* `Reg_Post`-objekt
- *ett* `Timer`-objekt
- *en* `int` som inneholder dagens dato (på formen ÅÅÅÅMMDD)

Programmet *skal* (i hvert fall) inneholde de ti klassene med (minst) sine datamedlemmer:

1. **`Hotell`** – inneholder:
    - navn, gateadresse, postadresse, mailadresse, filnavn (`char`-pekere)
    - telefon, fax, frokost-pris, extra-seng-pris og antall-fasciliteter (`int`'er)
    - array med `char`-pekere til teksten/beskrivelsen av fascilitetene (f.eks. Basseng, Konferanserom, Innendørs parkering, Bredbånd, Bar, …..)
    - peker-array til listene for romkategoriene (`Singel`, `Dobbel`, `Suite`)

2. **`Rom`** – inneholder:
    - romnummeret (`int`) – sortert på/etter dette
    - en sortert liste med reservasjonene utført på/i dette rommet
    - antall senger (`int`) i dette rommet
    - om frokost er inkludert eller ei (`bool`)

    Denne klassen har tre subklasser

    3. **`Singel`** – inneholder:
        - (ingen andre data – utover de arvet fra `Rom`)

    4. **`Dobbel`** – inneholder:
        - om rommet har mulighet for å inneha *en* extra seng (`bool`)

    5. **`Suite`** – inneholder:
        - antall senger pluss/minus (`int`) ift. de standard fire som *alle* suiter inneholder
        - antall kvadratmeter (`int`)
        - en "lengre" tekst med beskrivelse (`char`-peker)

6. **`Reservasjon`** – inneholder:
    - ankomst- og avreisedato (`int`'er) – sortert på det førstnevnte
    - antall døgn (dvs. forskjellen på ankomst- og avreisedato) (`int`)
    - array (`float`) med prisen for de døgnene reservasjonen gjelder
    - FIFO-liste med regninger
    - status for evt. extra seng – om den er tilgjengelig (0 eller 1), om så er tilfelle, er den i bruk (2) eller ei (1) (`int`)
    - antall beboere på rommet (`int`)
    - array med `char`-pekere til navnene for/på disse beboerne

7. **`Regning`** – inneholder:
    - teksten evt. nummeret for hva regninger gjelder (`char`-peker)
    - dets totalsum (`float`)

For å oppsummere datastrukturen hittil (som dere *skal* bruke): Et `Hotell`-objekt inneholder lister av `Rom`-objekter (i praksis `Singel`, `Dobbel` og `Suite`). Hvert `Rom` igjen inneholder en liste med `Reservasjon`s-objekter. Nåværende beboer (`Reservasjon`) på et rom kan igjen inneholde en liste med `Regning`-objekter.

Objekter av typene `Rom`, `Reservasjon` og `Regning` de eneste som skal legges inn i lister, og derfor de eneste som trenger å arve noe fra LISTTOOL.

Dataene for *alle* objektene 1-7 leses fra filen: < FORKORTET HOTELLNAVN>.DTA
(<FORKORTET HOTELLNAVN> er å finne på filen HOTELLER.DTA. Mer nedenfor …)

Det tre neste (og siste?) klassene er ”ressurs”-objekter som det finnes kun *ett* stykk av i programmet. De to første (`Timer` og `Reg_Post`) er globale, mens et `Pris`–objektet oppstår lokalt de gangene det er bruk for et slikt.

8. **`Timer`** – inneholder:
    - Dataene her er uinteressante. Det er de tilgjengelige *funksjonene* som er viktige – og som dere *skal* bruke. Hele klassen er ferdiglagd av faglærer, og ligger på PROSJEKT-katalogen.

9. **`Reg_Post`** – inneholder:
    - antall standard/de mest vanlige regningsposter (`int`)
    - `char`-array inneholdende tekstene for disse mest vanlige postene (f.eks. Pay-TV, Internet, telefon, minibar, bar, restaurant)

    Dataene for dette objektet leses fra filen: REG_POST.DTA (mer nedenfor)

10. **`Pris`** – inneholder:
    - standard/vanlig pris for overnatting i uka (søndag kveld - fredag morgen) (`int`)
    - standard/vanlig pris for overnatting i helga (fredag kveld - søndag morgen) (`int`)
    - en eller annen ”datastruktur” for å representere resten av fildatene for spesielle priser/tilbud i egne tidsperioder og for ulike romtyper (mer nedenfor).

    Dataene for dette objektet leses fra filen: <FORKORTET HOTELLNAVN>.PRS (mer nedenfor)

Tips: *Tegn opp, og bli ordentlig sikker på hvordan datastrukturen er/ser ut, og hvordan den fungerer ifm. de ulike funksjonene dere skal lage* (angitt i ”Innledningen” og ”Menyvalg/funksjoner”). Se også avsnittet ”Delinnlevering nr.1” (siste side).

Menyvalg / funksjoner

Dere skal lage et fullverdig program som har følgende muligheter/menyvalg:

1. **B - Bestill/reserver/book et rom**
   Det spørres først om en romtype (Singel, Dobbel eller Suite). Deretter spørres det etter ankomstdato (som *må* være >= dagens dato) og avreisedato (som *må* være større enn dette igjen). Så spørres det om vedkommende vil ha frokost, evt. har behov for ekstra seng (gjelder *kun* Dobbelt-rom). Det forsøkes så å foreta en reservasjon av et egnet ledig rom i aktuell tidsperiode, *startende på et tilfeldig romnummer*, i vedkommende romkategori. Lykkes dette, så spørres det etter navnene til rommets beboer(e). *Minst* ett må angis, som da blir ”reservatøren/bestilleren/bookeren”.

2. **A - Avbestill et rom**
   Programmet spør etter reservatørens navn. Deretter skrives/presenteres *alle* denne personens reservasjoner på skjermen. For hver må brukeren svare ”ja”/”nei” til om den aktuelle skal fjernes/slettes.

3. **I - Innsjekking på (ankomst til) hotellet**
   Programmet spør etter reservatørens navn. Det letes så etter *alle* rom som vedkommende har bestilt på dagens dato, og deres romnummer skrives på skjermen. For hvert rom blir det evt. spurt etter navn(ene) på resten av rommets beboer(e).

4. **U - Utsjekking (avreise) fra hotellet**
   Det spørres etter et romnummer. Om dette rommet har en reservasjon med avreisedato i dag, så skrives alle dets data til skjermen, inkludert en enkel "faktura" med prisen for: overnattingen de N døgnene, evt. extra seng med/uten frokost, evt. summen for regningene, samt totalsummen for alt dette. Før reservasjonen slettes fra hukommelsen, så skrives dets data (inkl. evt. regninger) *bakerst* (vha. "append") på filen:
   <FORKORTET HOTELLNAVN>.HST
   Tips: Gjenbruk (vha. parametre) den samme koden som brukes for å skrive til <FORKORTET HOTELLNAVN>.DTA

5. **R - Registrer/legg inn en regning på et rom**
   Det spørres etter et romnummer. Om noen bor på dette rommet for øyeblikket, så spørres det etter hva regningen gjelder og dets beløp. Et nytt `Regning`-objekt legges inn bakerst i listen hos reservasjonen.
   Når det gjelder "hva regningen gjelder" så kan brukeren skrive nummeret på en del standard valg som `Reg_Post`-objektet presenterer/viser på skjermen *eller* en fri tekst.

6. **Endre (om mulig) en bestilling/reservasjon/booking:**
   - **E 1 – Endre ankomst- og/eller avreisedato *før* innsjekking på hotellet**
   - **E 2 – Endre avreisedato når allerede bor (er innsjekket) på hotellet**
   - **E 3 – Bytte til et annet (ledig) rom enn det tildelte**

7. **Oversikt over:**
   - **O 1 - hoveddataene om *ett* hotell:**
     Alle dataene (se beskrivelsen av `Hotell`-klassen ovenfor), unntatt det om romkategoriene og deres rom, skrives på skjermen.
   - **O 2 - beskrivelser av *alle* hotellets suiter:**
     Alle dataene (unntatt reservasjonene) om *alle* hotellets suiter skrives ut.
   - **O 3 - *alle* reservasjoner som står i *en* persons navn:**
     Ber om en reservatørs navn. Skrives ut *alle* dennes reservasjoner.
   - **O 4 - når et rom er ledig:**
     Ber om er romnummer. Skriver hvilke perioder/tidsintervall rommet er ledig.
   - **O 5 - *alle* reservasjoner for/på et rom:**
     Ber om er romnummer. Skriver ut *alle* reservasjoner på dette rommet.

- **O 6 - *alle* data for *ett* roms nåværende beboer, inkludert dets regninger:**
  Ber om et romnummer. Skriver *alle* data om nåværende reservasjon inkl.regningene.
- **O 7 - *alle* ledige rom i *en* kategori en gitt tidsperiode:**
  Ber om en romkategori (Singel, Dobbel, Suite), ankomstdato og avreisedato. Skriver ut data om *alle* rom som er ledige i dette tidsintervallet.

8. **T - skriv alt om hotellet Til fil**
   Programmet skriver *alle* data om nåværende hotell tilbake til filen: <FORKORTET HOTELLNAVN>.DTA .

9. **H - bytte over til et annet Hotell**
   Gjør det samme som kommandoen 'T'. I tillegg spørres det om *fullt* navn på et annet hotell. På filen HOTELLER.DTA finner programmet nytt <FORKORTET HOTELLNAVN>, og leser så inn *alle* data om det nye hotellet fra *dets* .DTA-fil.

Alle slags feilsituasjoner (f.eks. ingen (ledige) rom i en kategori, ingen reservasjonen i en persons navn, rom har ikke avreisedato på nåværende dato, filer finnes ikke, ***m.m.***), og dertil egnede meldinger, er det ikke bemerket ovenfor. Dette må også selvsagt gjøres/kodes.

I tillegg må dere selvsagt lage `main` som "styrer hele butikken", samt funksjoner for å lese brukerens valg/kommando og en lengre utskrift med liste over lovlige valg/kommandoer. Og sikkert noen flere også ….. (f.eks. for å lese: tekster, "J"/"N", tall i et visst intervall).

Data til/fra filer

I programmet er det totalt involvert fem ulike (typer) filer:

1. **HOTELLER . DTA**
   Format: <FORKORTET HOTELLNAVN> <Fullt/helt hotellnavn>
   Det 1.feltet er *ett* ord. Mellom de to feltene ligger det *ett* blankt tegn. På resten av linjen ligger hotellets fulle/hele navn, som *kan* være på flere ord.
   Det er denne filen det letes på ifm. kommandoen "H". Brukeren oppgir altså det fulle/hele navnet for et hotell, og så letes det etter *dette* hotellets filnavn *før* ".DTA".

2. **<FORKORTET HOTELLNAVN> . DTA** **NB:** *En* slik fil *pr*.hotell
   Format: Dette må dere bestemme/finne på selv ……
   Filen skal altså inneholde *alle* data om et hotell, altså alt inni `Hotell`-objektet og dets datastruktur (dvs. *alt* om objekter fra klassene 1-7 beskrevet ovenfor).
   Denne filen oppdateres (leses inn fra/skrives ut til) ifm. kommandoene 'T' og 'H'.

3. **<FORKORTET HOTELLNAVN> . HST** **NB:** *En* slik fil *pr*.hotell
   Format: Noe likt med den rett ovenfor ……
   Filen inneholder *alle* (historiske) data om folk (fullbyrdede reservasjoner) som har sjekket ut fra *ett* hotell. Det skrives til den ifm. kommandoen 'U'.

4. **<FORKORTET HOTELLNAVN> . PRS** **NB:** *En* slik fil *pr*.hotell
   Utseende og format: Standard, uke:
   singel <pris>
   dobbel <pris>
   suite <pris>
   Standard, helg:
   singel <pris>
   dobbel <pris>
   suite <pris>
   Disse åtte faste linjene etterfølges evt. av en eller flere poster, hver med **formatet**:
   <Fra-dato> <Til-dato> <Ant. ganger det på linjen under repeteres>
   <"singel" | "dobbel" | "suite"> <"uke" | "helg" | "fast"> <Pris>

Alt dette leses inn i *ett* lokalt Pris-objekt *hver* gang en reservasjon utføres (kommandoen 'B'). "Hver gang" – fordi denne filen kan bli manuelt/dynamisk oppdatert hele tiden mens programmet kjører (ingen av de andre filene er slik).
Dere må selv bestemme hvordan "datastrukturen" inni `Pris` må være for å representere dette. `Float`-arrayen inni hvert `Reservasjon`-objekt skal inneholde "en kopi" av de korrekte prisene (som gjaldt *i det* reservasjonen ble foretatt) for *alle* døgnene reservasjonen omhandler.
Hint: Her blir det en del "småfikkel", så vent med å kode denne eksakte "prisberegningen". Legg til å begynne med bare inn faste verdier/priser for hvert døgn.

5. **REG_POST . DTA**
   Format: <Tekst for standard/mest vanlig regningspost nr.N>
   Filen inneholder en og en linje med tekstene for standard regningsposter, *f.eks.* "Pay-TV", "Telefon", "Minibar", "Bar", "Restaurant", "Internet". Alt på denne filen leses inn i det globale `Reg_Post`-objektet. Ifm. kommandoen 'R' blir disse presentert som en nummerert meny for brukeren. Hun/han kan da bare skrive ett av numrene (som referanse til dets standard tekst) eller en fritekst (dersom ingen av valgene beskriver det regningen gjelder).

Prosjekt / multifil-program

Dere *skal* utvikle hele dette programmet som et prosjekt, der programmet er splittet opp i flere ulike filer. Følgende (minst 13) .h-filer må lages:

- en med *alle* const'er (og evt. en med *alle* enum'er)
- en med deklarasjon av *alle* 'globale' funksjonsheadinger
- en *pr.klasse* med deklarasjon av dets innhold (datamedlemmer og funksjonsheadinger)
- listtool2.h (ligger allerede ferdig på PROSJEKT-katalogen)
- timer2.h (ligger allerede ferdig på PROSJEKT-katalogen)

Følgende (minst 13) .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)
- listtool2.cpp (ligger allerede ferdig på PROSJEKT-katalogen)
- timer2.cpp (ligger allerede ferdig på PROSJEKT-katalogen)

Hjelp: Se og lær av filene E19*.* på EKSEMPEL-katalog.

Annet (klargjørende?)

- LISTTOOL *skal* brukes ifm. løsningen av denne prosjektoppgaven. *Legg merke til og bruk LISTTOOL2.H og LISTTOOL2.CPP som ligger på PROSJEKT-katalogen.*
- **NB:** Definer alle globale variablene på samme fil som `main`. Når dere trenger å bruke disse på/i andre filer, så refererer dere til dem vha. `extern` …… i disse filene.
- Noen aktuelle `const`'er *kan* være: `NVNLEN, STRLEN, MAX_FASCILITETER, MAX_PAA_ROM, MAX_RESERVASJONS_DOGN, STORSTE_ROMNR, ….`
- Om et personnavn skrives med store/små bokstaver er likegyldig (ved sammenligning).
- Om dere kan/ønsker å bruke den standard string-klassen på <string> (side 303-310 i læreboka) er opp til dere selv.
- I dette programmet har vi bare skjært igjennom og sagt at "slik *er* og *virker* det bare på *alle* hoteller". F.eks er det ikke mulig med:
  - mer enn de tre typene rom (f.eks. finnes ikke familierom og to-mannsrom)
  - å bo på et rom i flere dager , borte en liten periode, og så komme tilbake til samme rom
  - variabelt antall personer som bor på et rom i tidsperioden
  - spesielle rabatter for enkelte kunder (større grupper, faste partnere)
  - egen bookingmodul av konferanserom som må matche med romkapasiteten
  - at rengjøringspersonalet melder inn i systemet om hvilke rom som er klare/rengjort
  - problemer rundt samtidige brukere av systemet (sentral booking, online på web)
  - og masse annet ……
- 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. Angi dette i så fall på et ark først i besvarelsen deres.

Forslag til rekkefølge på implementasjon

1. `Main` m/`les_kommando`, `skriv_meny` og omrisset av klassen 1-7 m/datamedlemmer.
2. Bestem .DTA-filens (nr.2 ovenfor) format, og lag en "dummy" fil som det leses inn fra.
3. Fra/til denne .DTA-filen med *ett* `Hotell` og *hele* dets datastruktur.
4. Kommandoene 'T' og 'H', inkludert lesing fra HOTELLER.DTA.
5. Kommandoene 'B' (vent med om tidsperioden egentlig er ledig) og 'A'.
6. 'I', 'U' (inkl. .HST-fila), 'R' (lag også `Reg_Post` og REG_POST.DTA) og 'O'.
7. Om en tidsperiode *egentlig* er ledig (gjenstående arbeid fra 'B'), `Pris` og .PRS-fila og til slutt: 'E'.

Innspill til gjøremål 1.arbeidsuka (18.-22.mars)

1. Sette seg inn i/lese nøye oppgaveteksten. Analyse av problemstillingen og datastrukturen.
2. Delta på de tre forelesningene i uke 12 (mandag, tirsdag og fredag).
3. Lage grupperegler (dokumentet som ligger ute er *innspill*, ingen *ferdig* kontrakt) og bestemme hvordan konfigurasjonsstyring/versjonskontroll skal foregå.
4. Sette datoer/tidsfrister for pkt.1-7 rett ovenfor.
5. Være godt i gang med design/pseudokode for relevant kode.
6. Kodet/implementert pkt.1 under "Forslag til rekkefølge på implementasjon" (ovenfor).
7. Overholde fristen for og innholdet i "Delinnlevering nr.1" (se rett under).

Delinnlevering nr.1

Innen fredag 22.mars 2013 kl.09.00 *skal* dere levere følgende (på papir) til faglærer:

- *Ett* A4- eller A3-ark med *detaljert* tegning av datastrukturen
- Navnet på gruppedeltagerne, inkl. kontaktinfo (navn, mail, mobil) for alle i gruppen
- Grupperegelene, signert av alle gruppens medlemmer. Se *innspill* til kontrakt på: http://www.hig.no/~ooprog/obliger/regler.pdf
- Individuelt signert bekreftelse fra *hver* av deltagerne, se: http://www.hig.no/~ooprog/obliger/bekreftelse.pdf

Delinnlevering nr.2

Innen tirsdag 9.april 2013 kl.09.00 *skal* dere sende (på mail til faglærer) *en* zip-fil innholdende *alle* prosjektets (minst) 26 .h- og .cpp-filer, relevante .DTA-filer og .exe-filen for programmet. Dere skal da *minst* ha gjort ferdig (kodet og fungerer korrekt) t.o.m. pkt.5 under "Forslag til rekkefølge på implementasjon" (ovenfor).

Sluttinnlevering

Følgende *skal* leveres (**innen tirsdag 23.april 2013 kl.10.00**) :

- Utskrift av programmet (*alle* .h- og .cpp-filer) i ett eksemplar.
- DVD/CD/USB-stick med *alle* filene som trengs for å kjøre programmet fra Visual Studio C++. Dvs. *alle* .dta-, .h- og .cpp-filer, *hele* katalogen og dets underkatalog "debug" der alle VS-filene ligger, samt en kjørbar (.exe-fil) versjon av programmet. **Testkjør dette selv fra der filene ligger, på en maskin/konto som *ikke* allerede inneholder prosjektet – men Visual Studio, før dere leverer.**
- Filene som det skrives til/leses fra (.dta) sitt format og eksempel på deres utseende.
- Egne presiseringer/forutsetninger (om dere har måttet foreta dette).

Gruppe(sam)arbeid

Sørg for at alle ytre rammer er lagt til rette for et godt og konstruktivt samarbeide. Dette gjøres best ved å sette opp klare og konkrete grupperegler (se eget notat med *innspill* til dette). *Jobb mye, effektivt og målrettet allerede fra første stund* (dvs. start "langspurten" tidlig, se "Innspill til gjøremål 1.arbeidsuka"). Og: sørg for å være "i rute" ved delinnlevering nr.2 – ellers får dere en *knallhard* avslutning.

Generelle krav til obligatoriske arbeider

Se: http://www.hig.no/~grprog/obliger#Gen_reg (ni punkter nederst på siden)

Lykke til!
FrodeH