Prosjektoppgave

i «IMT1082 - Objekt-orientert programmering» våren 2018

Frister:	***Mandag 16.april 2018 kl.09:00***
	NB: *Fredag 16.mars kl.09:00 (1.delinnlevering)*
	Torsdag 5.april kl.09:00 (2.delinnlevering)
Arbeidsform:	*Gruppe (tre (evt. to) personer – flere er ikke relevant)*
Arbeidsinnsats:	***Mye***

Innledning

Dere skal i denne prosjektoppgaven lage et litt større program som holder orden på:

- spillere som deltar i ulike idretter (*en* spiller kan delta på lag i flere ulike idretter)
- ulike (lag)idretter som er inndelt i ulike divisjoner/avdelinger med sine terminlister og tabeller. Eks. på dette kan være: fotball, håndball, ishockey og (inne)bandy.
- mange ulike divisjoner/avdelinger, der hver av dem består av et visst antall lag, lagenes navn/adresse og alle deres spillere, terminlisten for alle kampene mellom lagene, samt resultatene/målscorerne i de ulike kampene.

I hovedsak skal programmet håndtere følgende operasjoner:

- legge inn en ny spiller/idrett/divisjon
- fjerne en spiller/idrett/divisjon
- skrive data om *alle* spillere/idretter, *en* spiller/idrett/div. eller *alle* spillerne på et lag
- skrive (til skjerm eller fil):
    - terminliste for *en* divisjon/avdeling
    - *alle* kamper en gitt dato for en idrett/divisjon
    - tabellen(e) for en *hel* idrett eller *en* divisjon/avdeling
- lese inn (mange) kampresultater fra fil
- endre/redigere spillerne på et lag

Globale variable, klasser (og *litt* datastruktur)

Programmet *skal* inneholde bare to globale variable: ett objekt av klassen `Spillere` og ett objekt av klassen `Idrettene` (+ evt/frivillig ett objekt av klassen `RobustIO`). Programmet er basert på (minst) syv ulike klasser og deres datamedlemmer:

1. **`Spillere`**: Inneholder *alle* spillerne (`Spiller`-objekter) og et heltall som angir det siste fortløpende unike nummeret en spiller har blitt tildelt.
2. **`Spiller`**: Inneholder et unikt nummer for spilleren (som de sorteres etter), samt pekere til tekster med vedkommendes navn og adresse.
3. **`Idrettene`**: Inneholder *alle* idrettene (`Idrett`-objekter).
4. **`Idrett`**: Inneholder dets navn (sortert etter), *alle* divisjonene/avdelingene (`DivAvd`-objekter) i vedkommende idrett, samt en variabel (enum) som forteller hva slags tabelltype som brukes i vedkommende idrett (mer om dette under ”Annet”).

5. **`DivAvd`**: Inneholder dets navn (sortert etter), et heltall som angir antall lag i vedkommende divisjon/avdeling, en array (30 lang) med pekere til alle disse lagene (`Lag`-objekter) samt en to-dimensjonal matrise som angir terminlista for divisjonen/avdelingen. Denne matrisen består av pekere til `Resultat`-objekter.
6. **`Lag`**: Inneholder pekere til dets navn og postadresse/-sted, et heltall med antall spillere på laget, samt en heltalls-array (50 lang) med det unike nummeret til lagets spillere.
7. **`Resultat`**: Inneholder en char-array (9 lang) med datoen (på formen 'åååammdd') for kampen, to heltall som evt. angir resultatet i kampen (antall hjemme- og bortemål), en boolsk variabel som angir om dette er resultatet etter normal spilletid eller etter en tidsforlengelse/straffer, samt to arrayer (50 lang) som angir nummeret på spillerne som scoret de ulike målene for henholdsvis hjemme- og bortelaget.

Ekstra/frivillig:

8. **RobustIO**: Inneholder *ingen* datamedlemmer, *kun* funksjoner for å sjekke at input fra brukeren er robust/fornuftig (mye mer om dette under "Annet").

`Spiller` er subklasse av `NumElement`, mens `Idrett` og `DivAvd` begge er subklasser av `TextElement`. *Alle* de andre klassene er *ikke* subklasser av noe i LISTTOOL.

NB: Tegn opp, og bli ordentlig sikker på hvordan datastrukturen må være (ser ut), og hvordan den fungerer ifm. de ulike funksjonene dere skal lage (angitt nedenfor). Se også avsnittet "Delinnlevering nr.1" (mot slutten av dette dokumentet).

Menyvalg / funksjoner

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

1. **S A | <nr> | <navn> - skriv Alle Spillerne eller spiller med <nr> / <navn>**
   Brukeren skal her altså gi kommandoen 'S' etterfulgt av 'A' eller et spillernummer/-navn. Ut fra dette skrives/displayes *alle* data om *alle* spillerne eller den *ene* spilleren angitte via nummeret eller spilleren(e) med dette navnet.

2. **I A | <navn> - skriv Alle Idrettene eller idretten med <navn>**
   Brukeren skal her altså gi kommandoen 'I' etterfulgt av 'A' eller en idretts navn.
   Ved 'A' skrives alle idrettenes navn, antall divisjoner/avdelinger i hver og tabelltypen. Ved <navn> skrives det samme, men *i tillegg* alle dets divisjoners/avdelingers navn, antall lag i hver, navnene/adressene til disse lagene samt antall spillere på hvert lag.

3. **N S | I | D - Ny Spiller, Idrett eller Divisjon/avdeling**
   Brukeren skal her altså gi kommandoen 'N' etterfulgt av 'S', 'I' eller 'D'. Ut fra dette legges det inn en ny spiller, idrett eller divisjon/avdeling (under en idrett som brukeren spørres om). Duplikate idretter eller divisjoner/avdelinger (under samme idrett) får *ikke* forekomme. Mens duplikate spillere må gjerne opptre (de skilles jo via nummeret). Når det gjelder selve dataene om en ny divisjon/avdeling så skal *alt* dette leses inn fra fil, se tilfelle nr.1 under overskriften "Data til/fra filer" nedenfor.

4. **F S | I | D - Fjern Spiller, Idrett eller Divisjon/avdeling**
   Brukeren skal her altså gi kommandoen 'F' etterfulgt av 'S', 'I' eller 'D'. Ut fra dette fjernes/slettes en spiller, idrett eller divisjon/avdeling (under en idrett som brukeren spørres om). Ved fjerning av en idrett eller divisjon/avdeling bør brukeren få et kontrollspørsmål med om vedkommende *virkelig* ønsker å gjøre dette.

5. **L - skriv terminListe for *en* gitt divisjon/avdeling til skjerm eller fil**
   Brukeren spørres etter navnet for en idrett og en divisjon/avdeling. Deretter spørres vedkommende om et filnavn. Terminlisten for denne divisjonen/avdelingen skrives enten til en fil med dette navnet eller til skjermen (om brukeren svarte *kun* med enter/linjeskift). Terminlisten skrives ut som en to-dimensjonal tabell, der datoene skrives på formen "dd/mm".

6. **K - skriv (resultatet av) *alle* Kampene en gitt dato for en hel idrett eller en divisjon/avdeling til skjerm eller fil**
   Brukeren spørres etter navnet for en idrett. Vedkommende spørres også etter navnet på en divisjon/avdeling i denne idretten, men dette kan gjerne besvares blankt (kun enter). Det spørres også etter et filnavn, som også gjerne besvares med blankt (kun enter). Til slutt spørres brukeren om en dato på formen "ååååmmdd". Ut fra alt dettte skrives *alle* kampene på denne datoen for vedkommende idrett (eller kun dets ene divisjon/avdeling om brukeren ikke svarte blankt på dette) til enten fil eller skjerm (alt ettersom hva brukeren svarte). For alle kamper som allerede inneholder et resultat på denne datoen, så skrives (selvsagt) resultatet ut også.

7. **T – skriv Tabell(er) for en hel idrett eller en divisjon/avdeling til skjerm eller fil**
   Brukeren spørres om etter navnet for en idrett. Vedkommende spørres også etter navnet på en divisjon/avdeling i denne idretten, men dette kan gjerne besvares blankt (kun enter). Det spørres også etter et filnavn, som også gjerne besvares med blankt (kun enter). Tabellen(e) for hele denne idretten, eller kun den angitte divisjonen/avdelingen skrives så til skjerm eller fil (om dette er angitt). Tabellen(e) må beregnes ved å gå gjennom *alle* en divisjons/avdelings resultater og gi poeng alt ettersom hvilken tabelltype idretten er av.

8. **R – lese Resultatliste inn fra fil**
   Dette er beskrevet som tilfelle nr.2 under overskriften "Data til/fra filer" nedenfor.

9. **D – Data om alle spillerne på et lag**
   Brukeren spørres etter navnene til: en idrett, en divisjon/avdeling og et lag. Om noe av dette ikke eksisterer, så kommer det en feilmelding om det. I motsatt fall skrives *alle* data (nummer, navn og adresse) om *alle* spillerne på laget ut på skjermen.

10. **E – Endre/redigere (spillerne på et lag)**
    Brukeren spørres etter navnene til: en idrett, en divisjon/avdeling og et lag. Om noe av dette ikke eksisterer, så kommer det en feilmelding om det. I motsatt fall spørres brukeren om hun/han ønsker å legge til eller fjerne en spiller. Til slutt spørres brukeren om nummeret til denne spilleren, som så tas ut av/legges til laget.
    (**NB:** Dette er den eneste formen for redigering av de innlagte dataene som dere skal lage kode for i dette prosjektet. Andre aktuelle endringer *kunne* bl.a. ha vært: endre spilleres navn/adresse, legge til/ta vekk lag, editere terminlisten, editere på resultater (dato, mål, målscorere), endre lags navn/poststed. Men dette skal dere slippe ……)

Ekstra/frivillig:

11. **C – skriv 10-på-topp liste av toppsCorerne for en gitt divisjon/avdeling eller et gitt lag til skjerm eller fil**

    Brukeren spørres etter navnet for en idrett og en divisjon/avdeling. Deretter spørres vedkommende om et filnavn og et lagnavn (begge de to siste kan besvares med blankt). En "10-på-topp" liste med de som har scoret flest mål (antall og navn) i vedkommende divisjon/avdeling eller lag (om ikke besvart blankt) skrives så til fil (om dette er angitt) eller skjerm. Disse "10-på-topp"-listene må settes opp ved å gå gjennom *alle* resultatene, for en divisjon/avdeling eller et lag, og finne ut hvem som har scoret mest mål.

I tillegg må dere selvsagt lage main som "styrer hele butikken", samt funksjoner bl.a. for å lese brukerens valg/kommando og en lengre utskrift med liste over lovlige valg/kommandoer.

Data til/fra filer

Alle dataene som ligger inni de to globale variablene `Spillere` og `Idrettene` skal lagres på minst to filer: **SPILLERE.DTA** og **IDRETTENE.DTA**. Programmet må sørge for at alt dette automatisk leses inn ved oppstart og skrives ut ved avslutning av programmet. Bestem formatet selv.

Programmet skal også kunne lese inn opplysninger fra to andre filer:

1. alle data om en *hel* ny divisjon/avdeling (**filnavnet bestemmer brukeren**).
2. mange ulike kampresultater (fra filen **RESULTAT.DTA**).

I tilfelle nr.1 skjer denne innlesningen etter at brukeren har gitt kommandoen "N D", idrettens navn, navnet på den nye divisjonen/avdelingen, samt navnet til den filen der resten av opplysningene om denne divisjonen/avdelingen ligger. Denne filen inneholder opplysninger om alle lagenes navn, adresse, antall spillere, spillernes navn/adresse evt. *bare* nummer dersom vedkommende allerede finnes i blant `Spillere`-objektene, samt *hele* terminlisten for divisjonen/ avdelingen. Vi antar at det *ikke* finnes noen logiske eller formatmessige feil på denne filen.

I tilfelle nr.2 leses det automatisk fra filen **RESULTAT.DTA**, etter at brukeren har gitt kommandoen 'R'. Denne filen inneholder kampresultater fra *en eller flere* idretter, og fra en eller flere av divisjonene/avdelingene under hver idrett. For hver divisjon/avdeling kommer først en dato, og så ofte flere resultater på denne dagen (navn på lag som har spilt mot hverandre, kampresultatet, om dette er etter spillerforlengelse eller ei, samt et spillernummer for hvert eneste mål som er scoret). Her må dere selv finne fram til et fornuftig filformat som ivaretar alt dette.

Den store utfordringen i tilfelle nr.2 er at den kan inneholde logiske feil.

Vi begrenser oss til at følgende feil *kan* forekomme:

- navn på ikke-eksisterende idrett, divisjon/avdeling eller lag (hjemme-/bortelag).
- de to lagene har ikke spilt mot hverandre denne dagen.
- det er allerede registrert et resultat mellom disse to lagene.

Dere kan derimot få lov til å regne med at filen *ikke* inneholder bokstaver når det skal komme tall (eller omvendt), samt at formatet omkring selve kampresultatet (med mål og målscorerne) *alltid* er korrekt.

Tips: Det beste i tilfelle nr.2 er å lese filen **RESULTAT.DTA** *to* ganger. Den første gangen sjekkes det at filen ikke inneholder noen av feilene nevnt ovenfor. Om dette går bra, så leses filen en gang til og datastrukturen oppdateres. I motsatt fall får brukeren beskjed om det som var feil. Begge disse to innlesningene kan gjøres vha. de samme funksjonene, bare at man i de to tilfellene sender med en boolsk variabel som forteller funksjonene om de *virkelig* skal foreta selve oppdateringen i datastrukturen eller ei.

Prosjekt / multifil-program

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

- en med *alle* const'er (og en med *alle* enum'er)
- en med deklarasjon av *alle* 'globale' funksjonsheadinger
- en *pr.klasse* med deklarasjon av dets innhold (datamedlemmer og funksjonsheadinger)
- ListTool2B.h (ligger allerede ferdig på PROSJEKT -katalogen)

Følgende (minst 10) .cpp-filer må lages:

- en som inneholder main og definisjon av de to/tre globale objektene
- *minst* en fil som inneholder definisjon (innmaten) av *alle* de 'globale' funksjonene
- en *pr.klasse* med definisjon av klassens funksjoner (deres innmat)
- ListTool2B.cpp (ligger allerede ferdig på PROSJEKT -katalogen)

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

Annet (klargjørende?)

- ListTool *skal* brukes ifm. løsningen av denne prosjektoppgaven.
  Legg merke til og bruk 'ListTool2B.H' og 'ListTool2B.CPP' (se rett ovenfor).

- Definer de to/tre globale objektene 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, MAXLAG, MAXSPILLERE, DATOLEN,` ……

- Ulike idretter fungerer etter ulike tabellformer. Noen gir 2 poeng for seier, 1 for uavgjort og 0 for tap. Andre gir 3 poeng for seier, 1 for uavgjort og 0 for tap, mens noen gir 3 poeng for seier, 2 poeng for seier på overtid/straffer, 1 for uavgjort ved fulltid og 0 for tap. I programmet er det disse tre dere skal ta hensyn til. Dette gjelder derfor ifm. med ny idrett (kommandoen "N I") og utregning/utskrift av tabeller (kommandoen 'T').
  Så om dere derfor bør bruke subklasser (av `DivAvd`) og virtueller for å programmere dette, eller løse det på en annen måte må dere selv ta stilling til.

- For å forenkle programmet litt, så lar vi det være slik at hvert mål/poeng *alltid* har *en* målscorer. Dvs. vi tar f.eks. *ikke* hensyn til at i ishockey regner man også med assists, i basketball scorer man både 1, 2 og 3 poeng ad gangen, mens i volleyball angir man seieren som settsifre uten poengscorere.

- Om dere kan/ønsker å bruke den standard string-klassen på <string> (se side 303-310 i læreboka) er opp til dere selv. Men det er nok ikke alt som er like lett å få til av robust IO (se punktet like nedenfor) om dere velger å lage/kode klassen `RobustIO`.

- Skal programmet ikke sløse mer med hukommelsen enn nødvendig, så bør ikke terminlista være en to-dimensjonal matrise med fast størrelse, men at den oppstår dynamisk ved run-time (vha. new). Dette innebærer at man må ha en peker til pekere som hver peker til en-dimensjonale arrayer med pekere til `Resultat`-objekter. Tips for dette står på side 474-479 i læreboka. Men, det er ikke noe krav om at dere skal kode det på denne måten. Det *er* lov å lage en fast to-dimensjonal matrise, selv om dette med peker til pekere med pekere til objekter er det mest elegante (og en knøttliten utfordring for de av dere som ellers synes at denne oppgaven blir for lett. ☺)

- Det er lurt å tidlig planlegge hvilken rekkefølge dere bør implementere dette i, og hva som kan gå parallelt, eller forutsetter at annet er laget/fungerer allerede.

- 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 en egen fil i innleveringen deres (se «Sluttinnlevering»).

- **Ekstra/frivillig:** Lag klassen `RobustIO`. Det lages kun *ett* objekt av denne klassen, og det blir liggende globalt sammen med `Spillere` og `Idettene`. Klassen er en verktøy-kasse som inneholder *kun* funksjoner (ingen datamedlemmer) for å sjekke/ garantere at *all* input/inntasting fra brukeren *virkelig* inneholder korrekte og fornuftige verdier. Disse brukes/tilkalles på *alle* relevante steder i koden. Den bør inneholde i hvert fall følgende funksjoner:

  - char* lesNyttNavn(char* t)
    Skriver ledeteksten `t`. Leser inn et nytt (bruker `new`) lovlig (bruker `okNavn` nedenfor) navn. Returnerer med en peker til dette navnet.
  - char* lesNyAdr(char* t)
    Skriver ledeteksten `t`. Leser inn en ny (bruker `new`) lovlig (bruker `okAdr` nedenfor) adresse. Returnerer med en peker til denne adressen.
  - bool okNavn(char* s)
    Returnerer true/false til om teksten `s` *virkelig* er et navn, som *kun* kan inneholde bokstaver, blanke og/eller bindestreker.
  - bool okAdr(char* s)
    Returnerer true/false til om teksten `s` *virkelig* er en adresse, som *kun* kan inneholde bokstaver, blanke og/eller sifre.
  - int tall(char* t, const int MIN, const int MAX)
    Skriver ledeteksten `t`. Returnerer et *tall* i intervallet [MIN, MAX]. Funksjonen *må* sikre at det *virkelig er* noe *numerisk* som blir skrevet inn.
  - char tilStor(char ch)
    Funksjonen returnere `ch` omgjort til stor bokstav, også 'æ', 'ø' og 'å'.
  - char* strip(char* s)
    Funksjonen returnerer med en peker til `s`, etter at blanke er tatt vekk i begge ender.

Grovt forslag til rekkefølge på implementasjonen

1. `Main` m/`lesKommando`, `skrivMeny` og omrisset av *alle* klassene m/datamedlemmer.
2. Bestem flest mulig `const`'er, skriv pesudokode og kod (globale) hjelpefunksjoner (jfr. siste avsnittet *rett* over tittelen «Data til/fra filer» ovenfor).
3. Bestem formatet for **SPILLERE.DTA**, **IDRETTENE.DTA** og etter hvert **NY_DIV.DTA**. Legg inn noen testdata på disse, og les dette inn i datastrukturen.
4. Implementer kommandoene: 'N', 'S', 'I', 'D' og 'E' (Hva som må gjøres i rekkefølge eller kan gjøres parallelt i dette og det foregående punktet, må dere selv finne ut/bestemme.)
5. Implementer kommandoene: 'F', 'L', 'K', 'T' og 'R' (+ evt. 'C') (Hva som må gjøres i rekkefølge eller kan gjøres parallelt, må dere selv finne ut/bestemme.) *Eksakt* format for utskrifter (til skjerm eller fil) må også bestemmes ifm. kommandoene 'L', 'K' og 'T'. Formatet for **RESULTAT.DTA** må bestemmes ifm. 'R'.

Gjøremål 1.arbeidsuka (12.-16.mars)

1. Gjøre pkt 1.1 og punktene 2.1-2.5 på websiden om prosjektet.
2. Sette seg inn i/lese nøye oppgaveteksten (jfr. pkt.5 på websiden om prosjektet). Analyse av problemstillingen og datastrukturen.
3. Delta på de tre forelesningene i uke 11 (mandag, onsdag *og fredag*).
4. Lage grupperegler (jfr. pkt.1.2 på websiden om prosjektet). **NB:** Dokumentet som ligger ute er *innspill, ingen ferdig* kontrakt.
5. Gjøre pkt.3 og 4 på websiden om prosjektet.
6. Bestemme datoene for når de ulike punktene under «Forslag til rekkefølge på implementasjonen» (rett ovenfor) skal være ferdig.
7. Utføre og avkrysse alt på sjekklisten (jfr. pkt.6 på websiden om prosjektet).
8. Overholde fristen for og innholdet i «Delinnlevering nr.1» (se rett under).

Delinnlevering nr.1

Innen fredag 16.mars 2018 kl.09:00 *skal* dere ha gjort følgende:

Levert *(på papir)* til emnelærer*:*

1. navnet på gruppedeltagerne, inkl. kontaktinfo (mail og mobil) for alle i gruppen.
2. *ett* A4- eller A3-ark med *detaljert* tegning av datastrukturen.
3. gruppereglene, signert av alle gruppens medlemmer.
4. individuelt signert bekreftelse fra *hver* av gruppedeltagerne (jfr. pkt.1.3 på websiden om prosjektet).
5. utført, avkrysset og signert sjekkliste (jfr. pkt.6 på websiden om prosjektet).

Dataene i pkt.1 legges også inn i README-filen for prosjektet på Bitbucket (fjern all den andre teksten Bitbucket foreslår skal være med der).
Dokumentene i pkt.2-3 ovenfor skal også (innscannet) legges inn i GruppeXX-mappen deres på Google Docs.

Delinnlevering nr.2

Innen torsdag 5.april 2018 kl.09:00 *skal* dere ha lastet opp (committed) deres siste versjon av koden i prosjektet. Dere skal *minst* ha gjort ferdig (kodet og fungerer korrekt) t.o.m. pkt.4 under «Forslag til rekkefølge på implementasjon» (ellers vil dere ligge veldig dårlig an … ☹).
I tillegg skal alle aktuelle testfiler i GruppeXX-mappen på Google Docs være oppdatert (jfr. pkt.2.6 på websiden om prosjektet). *Disse lages/fylles ut etter hvert som dere skriver pseudokode for hver kommando (altså **før** dere begynner å skrive selve koden).*

Sluttinnlevering

Innen mandag 16.april 2018 kl.09:00 *skal* dere ha:

- lastet opp (committed) deres fungerende, endelige og siste versjon av koden i prosjektet.
- lagt inn *minst* de obligatoriske testdataene i alle filene (jfr. pkt.7 på websiden om prosjektet).
- *Testkjørt prosjektet i god tid før fristen ved å clone det hele ned til en helt ny katalog, og kjørt det derfra.* (Dette blir en simulering av hvordan læringsassistentene vil teste/kjøre og oppleve programmet.)
- i GruppeXX-mappen (på Google Docs):
  - oppdatert og ferdigstilt (fylt ut komplett) alle testfilene.
  - lagt *en fil* med beskrivelse av *alle* filformatene og eksempler på filenes utseende.
  - evt. ha lagt *en* egen fil («readme.doc/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 best ved å sette opp klare og konkrete grupperegler (se pkt.1.2 på websiden om prosjektet, som er *innspill* til dette). *Jobb mye, effektivt og målrettet allerede fra første stund* (dvs. start «langspurten» tidlig, se «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://folk.ntnu.no/frh/grprog/obliger#Gen_reg (spesielt det åttende punktet)

Lykke til!

FrodeH